Using Linguistic Features to Improve Prosody for Text-to-Speech

Rose Sloan

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2023

Abstract

Using Linguistic Features to Improve Prosody for Text-to-Speech

Rose Sloan

This thesis focuses on the problem of using text-to-speech (TTS) to synthesize speech with natural-sounding prosody. I propose a two-step process for approaching this problem. In the first step, I train text-based models to predict the locations of phrase boundaries and pitch accents in an utterance. Because these models use only text features, they can be used to predict the locations of prosodic events in novel utterances. In the second step, I incorporate these prosodic events into a text-to-speech pipeline in order to produce prosodically appropriate speech.

I trained models for predicting phrase boundaries and pitch accents on utterances from a corpus of radio news data. I found that the strongest models used a large variety of features, including syntactic features, lexical features, word embeddings, and co-reference features. In particular, using a large variety of syntactic features improved performance on both tasks. These models also performed well when tested on a different corpus of news data.

I then trained similar models on two conversational corpora: one a corpus of task-oriented dialogs and one a corpus of open-ended conversations. I again found that I could train strong models by using a wide variety of linguistic features, although performance dropped slightly in cross-corpus applications, and performance was very poor in cross-genre applications. For conversational speech, syntactic features continued to be helpful for both tasks. Additionally, word embedding features were particularly helpful in the conversational domain. Interestingly, while it is generally

believed that given information (i.e., terms that have recently been referenced) is often de-accented, for all three corpora, I found that including co-reference features only slightly improved the pitch accent detection model.

I then trained a TTS system on the same radio news corpus using Merlin, an open source DNN-based toolkit for TTS. As Merlin includes a linguistic feature extraction step before training, I added two additional features: one for phrase boundaries (distinguishing between sentence boundaries and mid-sentence phrase boundaries) and one for pitch accents. The locations of all breaks and accents for all test and training data were determined using the text-based prosody prediction models. I found that the pipeline using these new features produced speech that slightly outperformed the baseline on objective metrics such as mel-cepstral distortion (MCD) and was greatly preferred by listeners in a subjective listening test.

Finally, I trained an end-to-end TTS system on data that included phrase boundaries. The model was trained on a corpus of read speech, with the locations of phrase boundaries predicted based on acoustic features, and tested on radio news stories, with phrase boundaries predicted using the text-based model. I found that including phrase boundaries lowered MCD between the synthesized speech and the original radio broadcast, as compared to the baseline, but the results of a listening test were inconclusive.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my advisor Julia Hirschberg, whose guidance has been crucial over the course of my PhD program. I would also like to thank the many project students who have helped me in my research, including Ritvik Shrivastava, Syed Sarfaraz Akhtar, Mariel Anne Werner, Bryan Li, Adaeze Adigwe, Sahana Mohandoss, Stephen Eisner, Shayan Hooshmand, Eleanor Lin, and Shivani Ghatge. Their contributions to my work have been invaluable. I would also like to thank my labmates: Sarah Ita Levitan, Erica Cooper, Victor Soto, Brenda Yang, Lin Ai, and Run Chen. I would like to thank Erica Cooper in particular, as her prior work on TTS (and extensive documentation of relevant tools) was extremely helpful. Finally, I would like to thank the many friends and family who have supported me over the past years.

Chapter 1: Introduction and Background

1.1 Introduction

Modeling prosody for the purposes of improving text-to-speech (TTS) is a crucial but difficult problem. Prosody is a broad term for elements of speech beyond the specific phonemes that are uttered. It includes aspects such as phrasing, accent, pitch, and rhythm. Synthesized speech with inappropriate prosody can sound highly unnatural to listeners or resemble a stereotypical inhuman "robot voice." Additionally, longer utterances with unnatural prosody can be difficult for listeners to follow and comprehend, especially if they contain few phrase breaks or oddly placed phrase breaks. Furthermore, in some cases, emphasizing inappropriate words or otherwise producing unnatural prosody can alter the meaning of an utterance or create ambiguity. (For example, the meaning of the sentence "I never said she stole my money" can vary depending on which words are emphasized.) Therefore, it is very important for a TTS system to output utterances with reasonable prosody. However, synthesizing appropriate prosody remains a difficult problem, as prosody is dependent on a number of contextual features, and there are many possible prosodic realizations for any given utterance.

Modern end-to-end TTS systems can learn natural prosody through seeing large amounts of data. While these systems can output highly natural speech, they require large training sets and therefore may not be suitable in domains where less data is available. Additionally, these systems are generally quite opaque and often do not have explicit prosody models, which can make it difficult to alter the prosody of a synthesized utterance. This can cause problems when trying to output utterances in a prosodically distinctive domain, such as radio news speech, or when trying to correct unnatural output.

In this thesis, I present work centered around the idea of improving the prosody of synthesized speech using a two-step process. In the first step, the locations of an utterance's phrase boundaries and pitch accents are predicted using a large number of linguistic features extracted from text. As this step uses only text-based features, it allows us to predict breaks and accents in novel utterances that we wish to synthesize. In the second step, these prosodic events are provided as input to a neural TTS pipeline, allowing the system to learn the acoustic properties of breaks and accents. These models can then properly synthesize breaks and accents in test utterances.

In chapter 2 of this thesis, I discuss work first presented at the Speech Synthesis Workshop in 2019 [1]. This chapter introduces a model for predicting prosodic events from text in radio news speech. This model was trained on the Boston University Radio News Corpus and uses a large variety of linguistic features. The model includes some features that have never been used before for prosody prediction, as well as some features that have only been used in older work that did not use modern NLP tools. In chapter 3, I present a modified version of this model suitable for predicting prosodic events in conversational speech. I present results of training on a task-oriented corpus of conversational speech (the Columbia Games Corpus) as well as results using a corpus of freeform conversational speech (the Switchboard Corpus). Additionally, I discuss the performance of these prosody prediction models in cross-corpus and cross-genre settings.

In chapter 4 of this thesis, I shift my focus to the second step of the process and discuss work first presented at Speech Prosody in 2022 [2], where I use prosodic events to synthesize prosodically appropriate speech. I present a prosodically enhanced Merlin pipeline that uses predicted phrase boundaries and pitch accents to improve the prosody of synthesized speech. I compare news stories synthesized using the prosodically enhanced pipeline to those of the standard Merlin pipeline, looking at both objective metrics and the results of a subjective listening test. Finally, in chapter 5, I present newer work using the DCTTS model for end-to-end TTS. I present the results of incorporating phrase boundaries into the input to DCTTS and discuss directions for further work with end-to-end TTS.

1.2 Prosodic Events and The ToBI Labeling System

Prosody assignment is, implicitly or explicitly, a part of all TTS systems, as the system must determine pitch, duration, and other prosodic aspects of each phone it synthesizes. As prosody is a broad term that includes many aspects of speech, there are a number of ways to conceptualize prosody and to model prosody for TTS. Here, I will focus on improving the placement of prosodic events, specifically phrase boundaries and pitch accents. Phrase boundaries occur between adjacent words and indicate the end of a complete intonational phrase. (Intermediate phrase breaks, which represent a weaker but still prosodically marked break between two words, are not considered here.) Pitch accents are word-level accents placed on words that are emphasized by a speaker. These words may be emphasized as part of the standard flow of speech or intentionally emphasized for pragmatic reasons. English has several different pitch accents, corresponding to the pitch of the word being emphasized.

The Tones and Break Indices (ToBI) system is a set of conventions commonly used to annotate prosodic events in speech [3]. It was first developed by linguists and computational linguists studying prosody in the early 1990s. The original version of ToBI was developed specifically for data in Mainstream American English, though today, many variations exist to annotate prosody in other dialects of English and in other languages. Since all corpora used in this thesis contain American English speech, all references to ToBI refer specifically to the original Mainstream American English version of ToBI.

ToBI uses four tiers to label prosody. One tier contains an orthographic transcription of the utterance, while another tier contains miscellaneous information, such as information about disfluencies or noise. The two tiers containing the bulk of the prosodic information are the breaks tier and the tones tier. The breaks tier contains an index representing the level of boundary between each pair of adjacent words. There are five possible break levels, ranging from 0 to 4. 0 represents two words that have been elided together, while 1 represents a standard boundary between two words. 2 represents a slight juncture between words, while 3 represents an intermediate phrase boundary. Finally, 4 is the strongest break level, indicating a full intonational phrase boundary. (My work largely focuses largely on identifying these 4-level breaks.)

The tone tier is used to annotate pitch accents, phrase accents, and boundary tones. This tier marks which words are pitch accented, along with the type of each pitch accent. ToBI distinguishes between 5 types of pitch accent, including high (H*), low (L*), and several others. This tier also includes phrase accents, representing the pitch (high or low) between the final pitch accent in a phrase and the phrase boundary. Finally, it includes boundary tones, which indicate the pitch (high or low) at the end of an intonational phrase.

Expert annotators are needed to create ToBI labels. Therefore, there is relatively little data with manual ToBI annotations. In particular, even in corpora that contain ToBI annotations, often only a small portion of the corpus will be annotated. (In fact, in all three corpora used for prosody prediction in the following chapters, a large portion of the data in the corpus was not ToBI annotated.) One way to obtain additional ToBI-labeled data is to use AuToBI [4], a tool that predicts ToBI labels for aligned audio based on acoustic features. AuToBI is particularly useful for corpora that have been partially ToBI annotated, as it is possible to train AuToBI using the manually labeled data and then use it to predict ToBI labels for the remainder of the corpus.

1.3 Prior Work on Prosodic Event Prediction

The problem of predicting prosody from text has been studied since the mid-90s. Early work focused on using relatively simple features, such as part of speech, a word's position in a sentence or paragraph, and punctuation [5, 6, 7]. Since the early 2000s, there has also been work focused on using complex syntactic features, including supertags [8, 9].

Over the years, much of the work on prosody prediction has continued to focus on syntactic features. Ingulfsen showed that shallow features pulled from constituency and dependency parses outperformed more complex forms of syntactic chunking when performing phrase boundary detection [10]. Chen et al. looked both phrase boundary and pitch accent detection, using a neural model that incorporated part of speech features, as well as syntactic features related to whether words are at the boundaries of syntactic constituents [11]. Tepperman and Nava also focused on syntax, showing that a model built using parse tree transducers outperforms a simple n-gram based model for both phrase boundary and pitch accent detection [12]. More recently, Obin and Lachantin used a set of complex syntactic features extracted through deep syntactic parsing with the tree-adjoining grammar framework to predict phrase boundaries in both read and spontaneous speech [13]. Mishra et al. used part of speech tags and dependency relations and showed that they could substitute for using lemmatized word identity in predicting phrase boundaries, especially when using a context window corresponding to the length of an intonational phrase [14].

Other work has focused on semantic or pragmatic features, particularly in conversational speech. Yuan et al. examined differences in pitch accent between read speech and spontaneous speech, looking especially at differences in which function words accent, as well as looking at part of speech [15]. Nenkova et al. built a model to predict accent on conversational speech focusing on various pragmatic features, many related to information status [16]. Brenier et al. also examined various features provided with the conversational Switchboard Corpus, such as information status, animacy, and contrast and found that these features were generally less helpful than part of speech and word identity features [17]. Somewhat more recent work has focused instead on word embeddings. Rendel et al. finding that including GloVe and CBOW embedding features could greatly improve a model for pitch accent detection, though these feature only slightly improved a model for phrase boundary detection [18].

There has also been some work focused on developing neural models to predict phrase boundaries. Vadapalli and Gangashetty showed that RNNs and LSTMs outperform DNNs on phrase boundary prediction tasks [19]. Klimkov et al. used a combination of part of speech, syntactic dependency, and word embeddings to train a bidirectional LSTM model that could predict appropriate phrase boundaries in longer utterances [20]. Zheng et al. developed a biLSTM-based end-to-end model for predicting phrase boundaries that contained embedding layers as opposed to using pre-trained word embeddings [21].

Finally, some work has looked at phrase boundary detection in multi-corpus and multi-domain settings. Rendel et al. showed that a phrase boundary prediction model trained on a dataset of read speech collected for use in training TTS systems could generalize to a corpus of news data with little loss of precision but a large drop in recall (45% to 25%) [22]. Soto et al. investigated cross-language phrase boundary detection using acoustic features, and found that models trained on data from several languages performed better on test corpora than monolingual models [23]. Rosenberg et al. investigated phrase boundary detection on five different corpora from different domains, showing that performance on this task drops significantly in cross-domain settings [24].

1.4 Parametric TTS

Parametric TTS is a term used to refer to all approaches to synthesizing speech that involve using a machine learning model to predict the acoustic features of the synthesized speech. It is generally contrasted with concatenative approaches, in which synthesized speech is created by joining together segments found in the training corpus. Most early parametric TTS systems used HMMs to learn and predict the acoustic properties of each phone in an utterance. Ideally, HMM systems can learn a separate acoustic model for each phoneme in every possible linguistic and acoustic context, but due to data sparsity issues, in practice, similar contexts are clustered together during training. This generally leads to averaging effects and poor voice quality in the synthesized speech, and therefore, most HMM-based parametric systems produced poorer quality audio than most concatenative systems. Resultantly, until the advent of neural parametric TTS, the vast majority of commercial TTS systems were concatenative systems, not parametric systems.

Neural approaches for parametric TTS have been explored since the 90s [25, 26], but these approaches became much more common in the mid-2010s. Approaches from that time largely focused on using a deep feed-forward neural network to predict acoustic features [27, 28, 29], though there were some RNN-based or LSTM-based approaches [30, 31]. One notable system from this period was Merlin [32], an open source toolkit for DNN-based synthesis. I discuss Merlin more in chapter 4 of this thesis.

These early neural approaches generally included separate duration models, which predict the duration of each phoneme, and acoustic models, which predict the acoustic features of each frame of the output. These models also would not output waveforms or spectrograms directly, instead predicting acoustic features such as Mel cepstral coefficients and f0, and using an external vocoder to perform the actual synthesis. Furthermore, due to the fact that many of these systems (particularly the DNN-based systems) could not remember important contextual information, a large amount of features were extracted on the front end to provide additional contextual and linguistic information.

Modern systems, in contrast, are largely end-to-end systems. The most notable system of this type is Tacotron [33], released by researchers at Google in early 2017. End-to-end systems use an encoder-decoder architecture to take in a string of character or phonemes representing an entire utterance and output a spectrogram. Unlike earlier neural approaches, these models involve relatively little pre-processing or post-processing, as they do not use any manually extracted linguistic or acoustic features, instead using neural audio and text encoders. When these systems are combined with neural vocoders, they can produce very high quality speech. For example, Tacotron2 [34], which is often considered the state of the art, combines the Tacotron architecture with Wavenet [35], a neural vocoder. When trained on enough high quality data, Tacotron2 can produce highly natural-sounding speech, comparable to actual human speech.

One of the major drawbacks of using an end-to-end system like Tacotron is that they make heavy use of recurrent layers, making training and synthesis very slow. Therefore, most end-to-end systems that do not use a recurrent approach similar to Tacotron prioritize efficiency instead of strictly trying to improve voice quality over Tacotron. One system focused on efficiency is DCTTS, discussed more in chapter 5. Another drawback is that it can be difficult to control the output of end-to-end systems, as very minimal feature extraction is done.

There are advantages to using a parametric TTS system when attempting to improve prosody for text-to-speech. Unlike concatenative systems, which rely either on having segments in the training data that properly match the desired prosody or on post-processing concatenated audio, it is possible for a parametric system to automatically learn and reproduce the acoustic properties of prosodic events. In systems that are not end-to-end, it is possible to include linguistic features related to prosody in the pipeline. It is also possible to model F0 separately from other acoustic parameters, thereby training the system to produce appropriate pitch contours. While modeling prosody can be more difficult in end-to-end systems, a number of approaches exist, discussed below.

1.5 Prior Work on Prosody in TTS

Improving the prosody of synthesized speech has been a constant concern for TTS researchers. For modern TTS systems, there are several approaches to incorporating prosody. A few researchers have taken an approach similar to the one I propose and incorporate prosodic events directly into the front end of a TTS system. Malisz et al. used the Merlin toolkit for neural TTS and altered the front end to include the level of prominence of each word as a feature [36]. Fujimoto et al. worked with an end-to-end model for Japanese TTS and included markings for high and low accents into the front end [37]. It is worth noting that, unlike my work, neither of these papers proposes a method for generating prosodic features for novel utterances.

Other work has focused on incorporating linguistic features related to prosody into the front end of either RNN-based or end-to-end systems. Guo et al. demonstrated that including syntactic features in the front end of an end-to-end system can noticeably improve prosodic output, showing that extracted features pertaining to relationships between words were more effective than general features related to syntactic constituents [38]. There has also been various work indicating that including pre-trained word embeddings on the front end of TTS systems can improve output, with recent work in particular focusing on using BERT embeddings [39, 40, 41].

Finally, some approaches focus on training a prosodic embedding layer that affects the prosody of an entire utterance. As these embeddings represent the prosody of a full utterance, they do not directly capture individual prosodic events. The most notable of these approaches is Wang et al.'s Global Style tokens, which use a variational autoencoder to create "style embeddings" [42]. These embeddings can alter a speaker style, and in particular, they can output a novel utterance with the same prosody as a specific training utterance. Tyagi et al. present a similar approach but also provide a way to create style embeddings for novel utterances, based on BERT embeddings and syntactic distances between words [43].

Chapter 2: Prosody Prediction for News Data

2.1 Introduction

As presented in the previous chapter, the problem of predicting prosody from text has been studied since the 1990s. However, gaps in the literature remain. In particular, older work often uses relatively simple feature sets, focusing on easy to extract features such as a word's part of speech or position in its sentence. This is likely due to the fact that NLP tools at the time were less accurate, making it more difficult to extract any features that express more sophisticated syntactic or semantic information. While more recent work does incorporate complex features, many of these studies focus largely on one type of feature (such as focusing primarily on syntactic features or primarily on incorporating word embeddings). The most recent work has also primarily focused on phrase boundary detection, with less recent work on pitch accent detection. This leaves room for experimentation with less explored features and broader feature sets.

In this chapter, I present models for predicting phrase boundaries and pitch accents on radio news speech. This work was first presented at the Speech Synthesis Workshop in 2019 [1]. These models incorporate a large number and variety of linguistic features, allowing us to examine which features are most useful for these tasks. These models include features that are well known to be helpful for prosody prediction, such as syntactic features, as well as lesser explored features, such as specificity scores. We focus on radio speech for two reasons. First, a large amount of work has been done on the radio news corpus we use, allowing us to compare our results to prior work. Second, radio broadcasters are trained to speak with a distinctive prosody that holds listeners' attention and emphasizes key words. Training a model that captures this distinctive prosody can allow us to replicate it in future TTS applications.

In this chapter, I first present the corpus and the features used for our prosody models. Then, I present results for both the phrase boundary and pitch accent tasks. Finally, I analyze which features were most useful for each of the two tasks.

2.2 Corpus

The Boston University Radio News Corpus is a corpus of radio news data released in 1995 by Mari Ostendorf, Patti Price, and Stephanie Shattuck-Hufnagel [44]. It was created for the primary purpose of having a clean corpus that could be used to study prosody, particularly for speech synthesis applications. The corpus consists of two portions. The "radio" portion of the corpus consists of audio taken from news broadcasts on the radio station WBUR, all read by professional radio announcers and recorded in studio. The second portion, dubbed the "lab news" portion, also contains news stories read by professional radio announcers, this time recorded in a lab. The lab news portion of the corpus is much smaller than the radio portion and contains stories that also appear in the radio portion, so we do not use it in any experiments, in order to avoid inconsistency between the lab conditions and the studio conditions, as well as to avoid repeated utterances within the data.

The radio portion of BURNC contains seven hours of data from seven speakers, three female and four male. Most speakers have roughly one hour of data, though there is some variation, with speaker f3b having the most data at 107 minutes and speaker m3b having the least at 32. All news stories are split into short paragraph length utterances, generally no longer than three or four sentences. As these stories are all read by professional announcers in studio conditions (and, in some cases, are from edited stories), the data is mostly very clean, containing very little noise, disfluency, or other irregularity.

The corpus was released with gold standard transcriptions and part of speech tags. The transcriptions were created manually, while the part of speech tags were generated automatically and hand corrected. Additionally, ToBI labels are included for about an hour and a half of the corpus, including data from five of the seven speakers. This portion of the corpus was annotated manually with the ToBI labels. While we attempted to extend these labels to the rest of the corpus using

AuToBI [4], an automated tool for predicting ToBI labels from audio, we found that this slightly decreased performance of our models. This may be due to inaccuracies in the AuToBI labels or due to variation within the corpus. (For example, some data within the full corpus comes from radio announcers giving interviews, while the annotated portion only contains read news stories.) Due to these issues, we use only the hour and a half of manually annotated data in the experiments described in this chapter.

2.3 Model

We built models for two binary classification tasks: one to determine whether a given word comes before a 4-level phrase break and another to determine whether a word is pitch accented. ToBI makes finer grained distinctions, including five levels of phrase breaks and distinguishing between various types of pitch accents, such as H*, L*, !H*, and L*+H. However, as we are interested in having a model with high accuracy for synthesis purposes, we do not make these distinctions in any of our models, as they can be quite difficult for computers (and in some cases humans) to detect. (The differences between break levels below 4 in particular can be quite subtle.) Similarly, as we are only looking to place these prosodic events, not to make any specific predictions about pitch, we do not use phrase accents or boundary tones at all.

The model was trained on the ToBI-labeled BURNC data. This included data from five speakers, though the amount of ToBI-labeled data varied drastically among the different speakers, with two speakers (f2b and m1b) having the bulk of the data. Due to this unevenness, we could not use a standard leave-one-speaker-out cross-validation scheme. Therefore, in order to avoid overfitting to any one speaker's prosodic patterns, we instead held out all data from one speaker, f3a, as our test set, as this speaker had the smallest amount of ToBI-labeled data. There were 16,148 words in the training set from 87 different news stories. There were 692 words from 10 different news stories in the test set.

All models presented in this chapter were trained using a Random Forest model using 200 decision tree classifiers. We used this classifier, as we found it outperformed Support Vector Machine and Logistic Regression models trained using the same feature sets. We also briefly attempted to train neural models on a similar feature set that fully incorporated word embeddings as features. However, due to the small size of our data, these models did not outperform the Random Forest model. Additionally, the Random Forest classifiers allow us to find the feature importance for each feature, providing us with more information about what features are most useful for prosody prediction.

We trained this model on a large number of linguistic features, including lexical features, syntactic features, and many more. I describe these features in detail below, split into broad categories. A full list of all features used is provided in Appendix A.

2.3.1 Positional and Word-Level Features

The first features included in our model were simple positional features. Each word's position in its sentence (both its distance from the start of the sentence and its distance from the end of the sentence) was included in the model. The current sentence's length was also included as a feature.

Several simple word-level features were also included in the model. Specifically, the number of syllables in the current word was a feature, as was the punctuation (if any) appearing after the word. Additionally, named entity recognition (NER) was run on the corpus, using Stanford CoreNLP [45]. This tool determined whether each word was a named entity, and if so, tagged it as either a person, a location, or an organization. We then used the NER tags for the current and next words as features.

Finally, we used all Linguistic Inquiry and Word Count (LIWC) dimensions as features. LIWC is a dictionary-based tool that categorizes words into 73 dimensions [46]. These dimensions capture both semantic and syntactic properties, such as emotions, thinking styles, social attributes, and parts of speech. For example, the word *trying* belongs to five categories: cognitive processes, drives (i.e. needs and motives), verbs, words related to tentativeness, and words related to achievement. LIWC also has a function word category, which is highly relevant to our work, as function words are rarely accented.

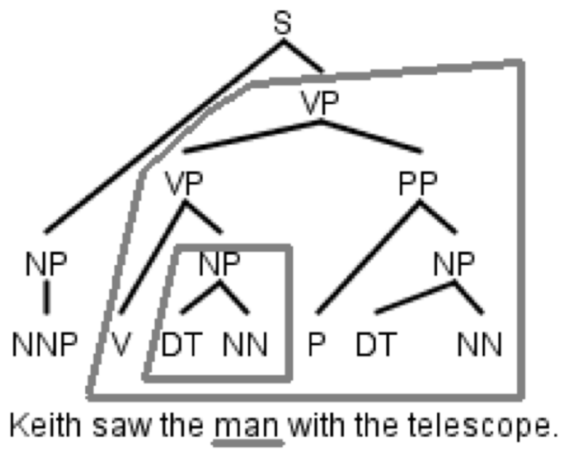

Figure 2.1: A constituency parse, of the sort used to extract syntactic figures in our model, for the sentence *"Keith saw the man with the telescope."* The smaller box indicates the smallest nontrivial constituent containing the word *man*, while the larger box indicates the minimal spanning tree between the adjacent words *man* and *with*.

2.3.2 Syntactic and Part of Speech Features

Part of speech tags, which were included as part of the original BURNC corpus, were included as features. Additionally, Stanford CoreNLP was used to get both dependency and constituency parses for each sentence. From the dependency parse, we included each word's syntactic function as a feature.

We extracted a large number of features from the constituency parse for each word. These features included the overall depth and width of the current sentence's parse tree, as well as the depth of the current word in the parse tree. We also looked at the smallest nontrivial constituent containing the current word. For example, in figure 2.1, the smallest nontrivial constituent containing *man* is the noun phrase *the man*. We included the length of this constituent and its label as features, along with the current word's position (both forwards and backwards) in the constituent. As a measure of syntactic distance between the current and next words, we also looked at the minimal spanning tree between these two words. For example, again looking at the word *man* in figure 2.1, the minimal spanning tree between *man* and *with* is the entire verb phrase *saw the man with the telescope*. We included this tree's width, depth, and label as features.

Another syntactic feature we included was supertags, based on early work showing that supertags were helpful in predicting prosodic events [8]. Supertags are similar to part of speech tags, but instead of assigning each word a part of speech, each word is assigned an elementary tree from the Tree-Adjoining Grammar framework [47]. This means that supertags convey more complicated syntactic information than part of speech tags. For example, supertags can distinguish between a noun in subject position and one in object position, and they can distinguish between a verb in active voice and one in passive voice. In order to get supertags, we used a Bi-LSTM-based supertagger that was pretrained on data from the Wall Street Journal. Each word's supertag (i.e. elementary tree) was included as a feature [48].

Finally, we examined a number of features that measured syntactic complexity at the sentence level. These features included number of clauses per sentence, number of dependent clauses, and number of complex noun phrases per clause. However, these features slightly decreased the performance of our model, so we omitted them from all models presented here.

2.3.3 Co-Reference Features

It is generally believed that an entity mentioned for the first time in speech is much more likely to be accented than one that has been previously mentioned. Therefore, we added several features related to co-reference. First, we used Stanford CoreNLP's co-reference resolution tool to find all co-references within the same news story. Based on these co-references, we then extracted a number of features. For any word with a co-reference that was a multi-word phrase, we used the head word of the phrase to get these features. For each word that had a previous mention, we included the number of previous mentions and the distance (in number of words) between the current word and its most recent mention. We also included the most recent mention's part of speech and syntactic function (as determined by the dependency parse). Additionally, we included the distance, part of speech, and syntactic function of the most recent explicit mention (i.e. a mention that was exact same token as the current word), if any existed. Similarly, we included the same three features for the most recent implicit mention (i.e. the most recent mention that was not the same token as the current word).

2.3.4 Word Embeddings

Prior work has shown that word embeddings are highly useful in predicting prosodic events. However, word embeddings can be difficult to use when not using neural models, as individual dimensions of word embeddings are not useful features on their own. Therefore, to reduce the word embeddings to one-dimensional features that would be useful in our Random Forest model, we performed k-means clustering on word embeddings over all words appearing in BURNC. We similarly clustered sentence embeddings, and included the cluster identities as features. Word embeddings were clustered into 5 clusters, while sentence embeddings were clustered into 20 clusters. (The number of clusters was chosen empirically based on model performance.) When using pre-trained embeddings, any unknown words in our corpus were excluded from the 5 clusters and assigned to their own cluster.

Table 2.1: Examples of words in each cluster, based on k-means clustering of BURNC using GLoVe embeddings pre-trained on Google News data.

Cluster ID	Words
0	*Bruins, Buffalo, children, women, hospital, weapons, drug*
1	*law, state, money, budget, educators, union*
2	*Boston, Mary, Chicago, Andrew, Peterson*
3	*say, see, giving, any, his, would*
4	*also, the, for, local, million*

Examples of words in each cluster, using word embeddings pre-trained on Google News data, are shown in table 2.1. This table shows that, while the distinctions between different clusters are not always perfectly clear, there are some apparent patterns. Here, cluster 0 contains some proper nouns, as well as a large number of specific nouns, many referring to broad groups of people such as *women* or *children*. Cluster 0 also contains many nouns with negative connotations such as *weapons* and *drug*. Cluster 1 contains words, mostly (but not exclusively) nouns, generally related to politics and finance. (Given that this was trained on news data, it makes sense that there would be enough such words to form their own cluster.) Cluster 2, which is the smallest of the clusters, contains almost exclusively proper names. Cluster 3 contains mostly simple verbs, along with some function words. Finally, cluster 4 contains mostly simple function words, along with some adjectives and numbers.

As there are many ways to obtain word embeddings, we tried a number of different embeddings, some trained directly on BURNC, others pre-trained on larger data sets. Among those pre-trained on BURNC, we first trained a set of 200-dimensional embeddings using the Word2vec skipgram model with negative sampling, with a window size of 4 words [49]. Additionally, since syntax is highly related to prosody, we trained another set of 200-dimensional embeddings using the same algorithm, but instead of using standard adjacency, we instead used a word's syntactic dependencies, as determined by the dependency parse.

For pre-trained embeddings, we first used a set of 300-dimensional GLoVe embeddings trained on roughly a billion words from stories on Google News [50]. We also looked at a set of pre-trained Word2vec embeddings that used syntactic dependency instead of linear adjacency; these were 300-dimensional embeddings trained on data from Wikipedia. Finally, we use a set of GLoVe embeddings that have been pre-trained on Wikipedia data and modified to remove all gender-related information. Although further work has indicated that these embeddings do not in fact fully remove gender markers [51], we found them useful for our tasks simply because they remove some high-level semantic information, not because of anything specific about gendered content. This may indicate that, in fact, these models could be improved by using some representation that captures the same syntactic and low-level semantic information contained in word embeddings but eliminates more high-level semantic information that is not particularly relevant to our tasks.

2.3.5 Speciteller

Speciteller is a tool for marking the level of specificity in a sentence [52]. Specificity here is defined at a sentence level, and it refers to the level of detail provided in a sentence. (For example, a sentence mentioning that an event happened in "several states" would provide less detail than one listing out specific states.) Speciteller gives each sentence a score between 0 and 1, with low scores assigned to vague sentences, such as those containing many pronouns and other vague words that can be used in a broad range of contexts (e.g. sentences referencing *people* instead of *students*), and higher scores assigned to more specific sentences, such as those containing proper names, specific quantities, and other narrow terms that can be used in relatively few semantic contexts. For example, the sentence *"Estimates vary widely on how much money could be saved"*, which has very little specific or concrete information, has a low score of 0.0186. In contrast, *"Quincy based Arbella Mutual Liability can now take over American Mutual's forty thousand car and home owner's policies"*, which has specific details including a specific quantity and several proper names, has a high score of 0.872. Each sentence's Speciteller score was included as a feature; the same value was assigned for all words in the same sentence.

2.4 Results

2.4.1 Phrase Boundaries

Feature Set	**Accuracy**	**F1**	**Precision**	**Recall**
Best Model	93.4%	0.810	0.810	0.810
Best Model without Punctuation	92.5%	0.774	0.798	0.752
Best Model without Syllables	92.0%	0.770	0.764	0.777
Best Model without Syntax	85.5%	0.575	0.580	0.570
Punctuation Only Baseline	91.2%	0.690	0.873	0.570
Majority Class Baseline	82.8%			

Table 2.2: Performance for **phrase boundary** detection on BURNC

The best performing model for predicting phrase boundaries used all syntactic features, all word-level features except for LIWC dimensions, the word embeddings trained directly on BURNC, and Speciteller scores. Table 2.2 displays the performance of this model, along with results when highly weighted feature groups were removed, as well as two baselines, one a simple majority class baseline that always predicts no break, the other a baseline that only uses punctuation. For each model, I have shown accuracy on the test set (which consists of a single speaker's data), along with F1, precision, and recall computed over the entire test set. With the exception of the baselines (which are at the bottom of the table), models are sorted from highest to lowest F1. Full results breaking down the impact of each feature group, as well as the most important individual features, are included in Appendix B.

As shown in the table, our best model has an accuracy of 93.4% and an F1 of 0.810. In terms of accuracy, this is over 2% better than the baseline that uses only punctuation. It also has a much higher F1 than that baseline, mostly due to poor recall (only 0.570) in the punctuation only model. This is due largely to the various syntactic features, as removing the syntactic features drops performance to only 85.5%. Other features that notably increase performance are word length (as represented by the number of syllables in the word) and punctuation after word. Our best model outperforms the strongest model in the literature: Rosenberg et al.'s model, which had an accuracy of 90.5% and an F1 of 0.781 [24]. It is also worth noting that 100% accuracy on this task (or any other prosody prediction task) is likely neither possible nor desirable. This is due to the fact that prosody assignment is not deterministic, so a model might place phrase breaks (or pitch accents) in prosodically appropriate positions but still deviate from the gold standard.

In our best model, precision and recall are equal, indicating that our model does not particularly predict too many or too few boundaries. Precision and recall also remain roughly equal when most features are removed. The primary exception to this is removing punctuation, which decreases recall much more than it decreases precision. This is unsurprising, as this feature is largely used to ensure that words followed by periods and commas are followed by phrase boundaries. (In fact, the fact that removing punctuation also reduces precision is more interesting, as it shows that the presence of commas that do not indicate phrase boundaries, such as in "Chicago, Illinois," does not particularly decrease precision.)

Unsurprisingly, given how much syntactic features increase the model's performance, the bulk of the heavily weighted features are syntactic features. Of the top five most heavily weighted features, four are syntactic, with the most heavily weighted feature being the width of the minimal spanning tree with the next word. The parse tree width, word depth in parse tree, and backwards position in smallest nontrivial constituent features were also highly weighted, ranking third, fourth, and fifth respectively. Other highly weighted syntactic features include the depth of the constituency parse, the width of the smallest nontrivial constituent, and supertag t3, which is the tree for nouns appearing at the end of noun phrases. In fact, only four of the twenty most heavily weight features were not syntax or part of speech features. These four features were Speciteller score in second, number of syllables in word in ninth, and the punctuation features for a period and comma appearing after the current word in tenth and thirteenth respectively.

2.4.2 Pitch Accents

Feature Set	**Accuracy**	**F1**	**Precision**	**Recall**
Best Model	81.9%	0.844	0.776	0.925
Best Model without LIWC	80.9%	0.832	0.778	0.895
Best Model without Embeddings	80.2%	0.826	0.771	0.889
Best Model without Syntax	79.1%	0.817	0.758	0.887
Content Words Baseline	79.9%	0.823	0.771	0.881
Majority Class Baseline	52.8%			

Table 2.3: Performance for **pitch accent** detection on BURNC

The best performing model for predicting pitch accents used all syntactic and word-level features, as well as gender-neutral embeddings, co-reference features, and Speciteller. Table 2.3 displays the performance of this model, along with results when heavily weighted feature groups are removed. Again, two baselines are shown, one a majority class baseline that always predicts no accent, the other a baseline that only uses the LIWC function word category as a features and therefore assigns accent to all content words. Again, accuracy, precision, recall, and F1 are shown, computed over the entire test set. Again, full results breaking down the impact of each feature group, as well as the most important individual features, are included in Appendix B.

This table indicates that our model outperforms the content words baseline by 2%, which is notable, as this baseline is already fairly strong. In particular, our best model has only slightly higher precision than the baseline (0.776 for our model vs. 0.771 for the baseline) but a much higher recall of 0.925 as opposed to 0.881 in the baseline, indicating that much of our model improvement comes from detecting function words that are in fact accented. (The discrepancy in both models between precision and recall, indicates that further improvement could come from improving detection of unaccented content words.) This model outperformed several prior models on this task (such as the model presented by Tepperman and Nava [12]), and it was only slightly worse than the best prior model, presented by Chen et al. [11], which achieved an accuracy of 82.7%.

Overall, recall is much higher than precision on this task on all models. Since this also true in the content words baseline (and in fact most of the improvement in the best model over that baseline comes from improvements to recall), this indicates that all our models struggle to de-accent content words that should not be accented. One reason for this may be that the corpus contains a large number of longer proper names or other compound nouns and is improperly accenting those names and compound nouns. For example, in a name like *Main Street*, only *Main* should be accented, and *Street* should be unaccented. However, the features that our model uses would often lead it to accent both words.

As our results show, once again, syntax features, taken as a group, are the most important set of features. In fact, without syntax features, the model performs worse than the content words baseline. Other highly important features include embeddings and LIWC features, the latter largely due to the function word category and similar categories like those for articles or prepositions. Interestingly, while co-reference features do improve the model, which fits with the popular wisdom that given words are de-accented, this improvement is slight. Removing all co-reference features only reduces the accuracy to 81.5% and the F1 to 0.840.

Removing any of the top feature groups substantially decreases recall. In fact, removing LIWC features actually increases precision slightly but decreases recall. This is unsurprising, because the model makes heaviest use of LIWC features like the function word category and the article category, which were used to identify content words to accent. Interestingly, removing syntax features, which overall reduces performance greatly, does substantially decrease precision. This may indicate that information about a word's syntactic role or position can indicate how likely it is to be accented, including indicating positions where content words should be unaccented.

The most heavily weighted feature in the best model is the function word category of LIWC, followed by the number of syllables in the current word, as content words and longer words are more frequently accented. Next, the third fourth, and fifth most heavily weighted features are respectively Speciteller score, width of the syntactic parse tree, and the sentence embedding cluster. Similarly, with one exception of the LIWC category for articles, the remainder of the top 20 most heavily weighted features are all either sentence embedding cluster IDs for surrounding sentences or syntactic features such as the word's depth in its tree, the width of the word's smallest nontrivial constituent, and whether that smallest nontrivial constituent is a prepositional phrase. These results, particularly given our model's high recall, indicate that in cases where function words are accented, this is due largely to syntactic context, as well as the syntactic and semantic features of the sentence or even paragraph in which the word appears.

2.4.3 Cross Corpus

To test how our model generalized to data outside of BURNC, we also tested it on Audix, another ToBI-annotated corpus of radio news data first released in 1993 as part of work on detecting pitch accents. Audix consists of ten AP News stories all read by the same female professional newscaster, though, unlike the portion of BURNC we used, these stories were all recorded under lab conditions. Of the ten stories, six have usable ToBI labels, for a total of 17 minutes of ToBI-labeled data. As this is a very small amount of data, we did not train any models on Audix and simply used it as a secondary test set.

The results of training on BURNC (still holding out f3a as a test set) and testing on Audix are shown in table 2.4 for phrase boundary detection and in table 2.5 for pitch accent detection. These models excluded embedding features, as they were clustered using the vocabulary in BURNC, but otherwise contained all features described in this chapter. Since this is a slightly different feature

Training Set				
			BURNC	Naive
Test Set	BURNC	Accuracy	92.6%	82.8%
		F1	0.783	
	Audix	Accuracy	90.1%	78.5%
		F1	0.739	

Table 2.4: Accuracy and F1 scores for cross-corpus evaluation of phrase boundaries between BURNC and Audix

Training Set				
			BURNC	Naive
Test Set	BURNC	Accuracy	80.9%	52.8%
		F1	0.776	
	Audix	Accuracy	80.6%	50.9%
		F1	0.772	

Table 2.5: Accuracy and F1 scores for cross-corpus evaluation of pitch accent between BURNC and Audix

set than the best models described above, the performance on the BURNC test set is provided for comparison's sake, along with a majority class baseline (labeled naive in the tables). Furthermore, while the precise feature set is slightly different, the most heavily weighted features (with the exception of embedding features for the pitch accent models) are largely the same as for the best models, with syntax features and Speciteller weighted highly in both models, and LIWC and word length weighted heavily for the accent model.

These results indicate that, for the most part, our models perform quite well on the Audix data. For pitch accent detection in particular, the drop in accuracy when testing on Audix is very small. For phrase boundaries, the drop is larger, though performance is still good. This is likely due to the fact that Audix contains a higher proportion of breaks than BURNC (perhaps due to the fact that all speech in Audix comes from a single speaker in lab conditions), leading to lower accuracy and recall. Overall, these results indicate that the best models trained on BURNC perform quite well on novel data, at least within the news genre, and are likely suitable for predicting the locations of prosodic events in other news corpora (and perhaps other read speech corpora), allowing ToBI labels to be used for synthesis or other applications.

2.5 Analysis

Based on these results, we can draw conclusions about the relationship between prosodic events and the various linguistic features we examined. First, we found that including positional features did not improve performance on either task. This result indicates that a word's precise position in a sentence is generally not predictive of whether it is accented or followed by a phrase boundary. While words at the end of sentences are almost always also at the end of phrases, this is already captured by punctuation features. Similarly, sentence length appears to affect both phrasing and accents, as the syntax tree width feature (which is quite similar to sentence length, varying only in cases where one word, such as a contraction, is split into multiple leaves in a syntax tree) is weighted heavily in both of our best models. Otherwise, a word's overall position in its sentence does not seem to affect phrasing or accents, once syntactic and punctuation features are accounted for.

In contrast, syntax and part of speech features were extremely helpful in both tasks. This is expected for the phrase boundary task, as boundaries often come at the end of large constituents and between two words that are syntactically distant. For pitch accents, there is clearly a link between part of speech and accent. (For example, it is unsurprising that the part of speech tag for nouns was weighted relatively highly, as nouns are often accented.) However, it is more surprising that many of the features related to the syntactic parse tree were weighted very heavily. This indicates that sentence length and syntactic complexity has a notable effect on accent, as well as a word's position within its constituent and the size of that constituent.

At the word level, we found that named entity recognition and word length were very helpful in predicting pitch accent and mildly helpful in predicting phrase boundaries. This is unsurprising, as long words and named entities are usually accented and often appear before boundaries. Conversely, punctuation was extremely helpful in predicting phrase boundaries and mildly helpful in predicting pitch accents. This is unsurprising, as periods and commas indicate phrase boundaries in many cases, and these boundaries are often preceded by accented words. LIWC was useful for the pitch accent task only, indicating that the categories captured by LIWC either have little relationship to phrasing or, in some cases, are simply better conveyed by other features (such as part of speech tags). For pitch accent, it is unsurprising that LIWC categories like function words, articles, and prepositions are useful features, as these words are often unaccented. Interestingly, the categories for drives, cognitive processes, and social processes were also weighted somewhat highly, indicating that some basic semantic information has an effect on accents.

Similarly, gender-neutral embeddings were the most helpful embedding features on the pitch accent task, indicating that some semantic information is useful for this task, but complex semantic information may be unhelpful. Furthermore, as embedding features (albeit using different embeddings) were useful for both tasks, these results confirm prior work that indicates word embeddings are useful for predicting prosodic events. However, since these results are based only on our clustered embedding features, it is hard to draw strong conclusions here without running experiments using neural models or other approaches that can fully use word embeddings.

Speciteller was a useful feature on both tasks, and, in fact, Speciteller was among the top five most highly weighted features in the best models for both tasks. These results indicate a relationship between specificity and prosody, as sentences with a high level of specificity tend to contain more phrase boundaries and pitch accents. This effect may be particularly pronounced within the news domain, as trained announcers are likely to emphasize words and pause for clarity when reading information-dense sentences. However, as the relationship between specificity and prosody has not been studied much, it is unclear how it varies across domains.

Finally, co-reference features were useful for the pitch accent model but not the phrase boundary model. This is unsurprising, as information status is believed to have an effect on accents but not on phrasing. However, as mentioned earlier, co-reference features were not particularly heavily weighted, and they only improved model performance slightly. This may indicate that information status is far less important than other syntactic and semantic features when determining which words are accented. It may also be an effect of using news data, where announcers tend to accent key terms, even those that have been previously mentioned within the same story.

2.6 Conclusions

In this chapter, I have presented strong models for predicting the locations of phrase boundaries and pitch accents in radio news speech. These models achieve relatively high accuracy, outperforming baselines that use commonly used simple heuristics (specifically, using punctuation to predict breaks and accenting content words). Additionally, the phrase boundary model outperforms the best model from the literature, and the pitch accent model achieves only slightly worse performance. Due to the high accuracy of these models, they can be used in applications that need predicted prosody labels. I will show how these models can be used to improve TTS systems in future chapters.

Furthermore, the results in this chapter provide more insight into which linguistic features have the strongest effects on prosody. We found that syntactic features were extremely useful for both tasks, and we additionally found that some semantic information is helpful for prosody prediction, especially for pitch accent detection. We also found that, while a word's information status does have an effect on whether or not it is pitch accented, this feature (and related features) appear to have relatively little (though positive) impact on the final model. Further work is needed to more clearly explore the relationship between information status and pitch accent on this corpus.

While our models are quite strong, there is still room for improvement, particularly on the pitch accent model. While the pitch accent model is reasonably accurate, its precision is not much higher than the (still fairly strong) content words baseline, indicating that it does not always identify content words that should be unaccented. (This weakness of the system is also indicated by the rather large gap between precision and recall.) One avenue for possible future improvement is to look at a broader set of semantic features. Additionally, it is possible to make more thorough use of word embeddings by using neural models that can use full word embeddings, as opposed to our clustering approach.

Future work could also explore models that make more specific prosodic predictions, particularly models that can predict intermediate (3-level) phrase boundaries and distinguish between different pitch accents. An accurate model to distinguish between different pitch accents could be particularly helpful for TTS applications, as different types of pitch accents sound quite different in speech. In particular, a model hat could distinguish between H* and L+H* would likely improve TTS output, as these are by far the most common types of pitch accents in English.

Chapter 3: Prosody Prediction for Conversational Data

3.1 Introduction

Most TTS systems are trained on read speech. However, there are a number of situations in which it may be desirable to replicate conversational prosody. As voice assistants and similar systems become pervasive, in order for them not to sound overly formal or stilted, it can be important for their TTS systems to output speech that more closely resembles the prosody of human conversations. Additionally, users who rely on screen readers may wish them to take on a more casual tone when reading more casual online speech (which, grammatically, often more closely matches conversational speech than formal writing).

There are many basic differences between read speech and conversational speech, including that conversational speech has many more fragments, disfluencies, and short utterances such as backchannels that must be accounted for when modeling its prosody. Additionally, some features, such as punctuation, that are extremely helpful when predicting prosody for read speech are not present in many conversational corpora. Therefore, there is a need for prosody models that are trained on conversational corpora and that use features relevant to those corpora.

In this chapter, I present models for predicting phrase boundaries and pitch accent from text on conversational speech. These models use a similar feature set to the models presented in the previous chapter, but a number of features have been adjusted to be more suitable for the conversational domain. I present models trained on a corpus of task-oriented conversational speech (the Columbia Games Corpus), as well as models trained on a corpus of open-ended conversational speech (the Switchboard Corpus). I also present cross-corpus and cross-domain results from these models.

3.2 Corpora

3.2.1 Columbia Games Corpus

The Columbia Games Corpus (henceforth referred to simply as Games) is a corpus of conversational task-oriented speech, created to study the prosodic differences between given and new information [53]. Speakers in this corpus played computer games in pairs. Due to the spontaneous and conversational nature of this data, it is much less “clean” than news data or other read data, as it contains many disfluencies, sentence fragments, and interruptions or overlaps between the speakers.

The games played in this corpus required the speakers to communicate with their conversational partners to achieve joint goals, and the tasks were designed so that accomplishing these goals generally required repeating the same nouns several times. In the first game, labeled the Cards task, both speakers saw computer screens with decks of cards. Each card depicted one to three items. Speakers could not see each other’s screens, and one speaker had to describe each of their cards one by one, while the second speaker had to find the matching card in their own deck. In the second game, labeled the Objects task, each player saw a screen displaying five to seven objects. These screens displayed the same objects for both speakers, but one object, designated the target object, was not in the same location on both screens. The goal of the game was for one speaker to describe the location of the target object on their screen so that the second speaker could place it as accurately as possible to make the two screens match. Each of these tasks was repeated several times per session, each time with different objects. The objects also varied in size and color.

The corpus consists of 12 sessions total, each from a different conversational pair. It contains 13 speakers total, all speakers of Standard American English, six male and seven female. 11 of the 13 speakers appear in two conversations, but they were paired with a different speaker for each conversation, and the two sessions were recorded on different days. Overall, each session (which includes both several iterations of the Cards task and several iterations of the Objects task) averages about 45 minutes in length. In all, the corpus has roughly 9 hours and 45 minutes of data and 73,800 words between all 12 sessions.

The corpus was orthographically transcribed and aligned manually. These transcriptions do not contain any capitalization or punctuation. They are segmented at the turn level. Additionally, over five hours of data, including all data from Objects tasks and roughly one third of the data from Cards tasks, was manually annotated with ToBI labels. The remainder of the corpus was automatically labeled with ToBI labels using an AuToBI model trained on the labeled Games data to predict ToBI features from the audio. The entirety of the Games Corpus was used in our experiments, using the manually labeled ToBI features when present and using using the automatically generated labels for the remainder of the data.

It is worth noting that the transcriptions of Games contain a large number of disfluency as well as repairs and restarts, making the text noticeably different from the corpora used to train most NLP tools. This means that the tools used to extract features are likely somewhat less accurate on this corpus than they were on BURNC. While we did experiment briefly using a parser trained to handle fragments, we found that this did not produce substantially better parses than CoreNLP, and other than adding in sentence segmentation (discussed below), we were able to use NLP tools on Games and get reasonably accurate (if imperfect) output without any further pre-processing.

3.2.2 Switchboard Corpus

The Switchboard Corpus is a large, well-annotated corpus of telephone conversations collected by Texas Instruments in 1991 and first released in 1997 [54]. The full corpus contains 2400 conversations from 543 speakers for a total of roughly 260 hours of data. Each conversation was roughly five minutes long. All speakers were native speakers of American English, and speakers from all regions of the United States were included in the corpus. Speakers were paired randomly and given one of 70 pre-generated topics, generally related to current events, hobbies, or some other topic of general interest. While some speakers participated in multiple conversations, no speaker was paired with the same conversational partner twice, nor given the same topic of conversation twice.

Due to the size of this corpus, it has been used for many different applications and has therefore been annotated with many features. It has also been manually transcribed multiple times. One transcript was released along with a portion of the Penn Treebank that contains parses for a portion of Switchboard. The other transcript in use is the MS State transcript, a transcript released by researchers at Mississippi State University in 1998. This transcript resegmented the corpus and provided more accurate and detailed transcriptions. It is worth noting that errors exist in both transcripts, and in particular, the time alignments of both transcripts have major inaccuracies. While these inaccuracies do not affect the results in this chapter, as we use only pre-labeled features and features extracted from text, they create challenges when using Switchboard for other text-to-speech applications. Ongoing work is being done to fix these alignments for future work on conversational TTS.

75 conversations in the Switchboard Corpus have been ToBI-labeled, though in some cases only part of the conversation was annotated. These labels are based on the MS State transcript, and therefore that is the transcript we use in all experiments in this chapter. The full data set used (including only utterances with usable ToBI labels) is 66,398 words long. Since our model also used some features that had been manually annotated using the Penn Treebank transcript, we pulled all data from the Switchboard in NXT release [55], a release of the corpus that attempted to consolidate both transcripts and all annotated features into a single XML format. Additional hand correction was performed to align sections where there were significant discrepancies between the two transcripts.

3.3 Model

The model presented here is largely similar to the one presented in the previous chapter, with some features modified to better suit the conversational corpora. Again, we trained two binary classification models, one to determine whether a word is followed by a phrase boundary and another to determine whether a word is pitch accented. (We continue to disregard any finer-grained distinctions in types of boundaries or accents.) All models were trained with a Random Forest classifier using 200 decision trees.

For determining the train-test split on Games, we were careful to avoid effects of entrainment, the tendency for conversational partners to develop more similar speaking styles over the course of a conversation. Therefore, we performed leave-one-speaker-out cross-validation where each fold used one speaker's data as the test set and all data from conversations where that speaker did not appear as the training set. (This differs from a standard leave-one-speaker-out approach, as it omits data from the test speaker's conversational partners from the training set.) For Switchboard, we are less concerned about overfitting to speakers, as there are many speakers in the data, each with a relatively small amount of data. Therefore, we simply hold out a random 20% of the data as a test set.

The features used in this model fall into the same categories as those used for news data. Modifications made to these features are discussed below. As with the models trained on BURNC, the full list of features used on the conversational corpora is available in Appendix A.

3.3.1 Positional and Word-Level Features

We once again used position in sentence (forwards and backwards) as well as sentence length as features. For Switchboard, sentence segmentation was done as part of the Penn Treebank transcript. For Games, however, no sentence segmentation existed, so we had to determine the sentence segmentation based primarily on pauses in the transcript. Based on patterns in the data, we determined that any silence of longer than 0.3 seconds was usually a sentence break, as long as it did not immediately follow a preposition, article, or conjunction nor immediately precede a conjunction. Therefore, we used this rule to segment the corpus into sentences. For both corpora, we also used position in turn and length of turn as features, and for Games, we used position in task and length of task. For Switchboard, we additionally included the current word's relative position in its sentence: the word's position in its sentence divided by the overall

Again, we used the number of syllables in a word as a feature, as well as all LIWC dimensions. For Switchboard, we used named entity recognition features, extracted using Stanford's CoreNLP toolkit [45]. These were omitted for Games, as there are very few named entities mentioned in the corpus, as all objects appearing in the tasks could be described using common nouns.

3.3.2 Syntactic and Part of Speech Features

For Switchboard, all conversations in our corpus were a part of the Penn Treebank release, which contains gold standard part of speech tags and constituency parses. For Games, these were extracted using CoreNLP, using the sentence segmentation rule discussed in the previous section. Additionally, dependency parses were extracted using CoreNLP for both corpora.

As with the news data, we used part of speech tags and syntactic functions as features. We also used the same syntactic features described in the previous chapter: the width and height of the constituency parse, the depth of the current word's leaf node, the width and label of the minimal constituent containing the current word, the position of the current word in that minimal constituent, and the height, width, and label of the minimal spanning tree between the current word and the next word. We also included each word's supertag, extracted using the same pre-trained supertagger as for the news data.

For Switchboard, we also included several additional syntax features, inspired by previous work from Hirschberg and Rambow [8]. These features include whether or not the current word is at the right edge of a major constituent and whether or not the current word is at the right edge of a coordinating constituent. We also included the distance in arcs between the current word and the next word in the constituency parse tree.

3.3.3 Co-Reference Features

Again, we used Stanford CoreNLP to extract co-reference features for both Games and Switchboard. As we were particularly interested in the role of co-reference features in Games, we hand-corrected some of co-references for Games (largely those in longer noun phrases). We included the same co-reference features that we used for BURNC. These were number of previous mentions, the distance to the most recent mention, and the part of speech and syntactic function of the most recent mention, as well as the same features for only implicit mentions and only explicit mentions.

3.3.4 Word Embeddings

Again, we were unable to use word embeddings directly in our random forest classifier, so we once again performed k-means clustering on the word embeddings of all words that appeared in the corpus to get 5 embedding clusters and then used cluster ID for the current word and surrounding words as features. We similarly clustered sentence embeddings into 20 clusters.

For Games, we looked at three sets of embeddings. First, we used a set of GLoVe embeddings trained on Twitter data [50]. While Twitter data does not perfectly match the domain of conversational speech (and has its own quirks specific to language on the Internet), the informal nature of Twitter means it is more similar to our conversational corpora than news data or Wikipedia data is. In order to account for this discrepancy, we also tried adapting the Twitter embeddings to the domain of the Games corpus, using an algorithm for mapping between two models for embeddings in the same language [56]. Last, we used the same gender-neutral embeddings we used for our BURNC model, as they were helpful in our BURNC pitch accent model, despite not being trained on news data. Gender-neutral embeddings may in fact be particularly useful on the Games Corpus, because, due to the task-oriented nature of the speech in Games, many of the words in the corpus with strong gendered connotations were objects appearing in the tasks and therefore, their specific semantics were not particularly important in context. (For example, one of the images appearing on screen during the tasks was of a mermaid.) For Switchboard, we only looked at the GLoVe embeddings trained on Twitter.

Cluster Id	**Sample Words**
0	whoops, nonsense, sorry, jeez, very, missed
1	yellow, reddish, four, wooden, vase, nail
2	gladiola, yello-, midum-, we'll, thin-
3	bends, tilted, dragged, dropped, covers
4	would, call, tell, dunno, give, want, went

Table 3.1: Samples from word clusters of Twitter GloVe embeddings domain-adapted to the Games Corpus

Examples of words from each cluster, when using domain-adapted embeddings on Games, are shown in table 3.1. Cluster 0 contains a large number of interjections as well as some other common words. Cluster 1 includes adjectives and many nouns corresponding to items (such as *vase* and *nail*). Cluster 2 contains a large number of interrupted partial words. Cluster 3 largely include verbs, while cluster 4 largely includes function words.

3.3.5 Speciteller

As with BURNC, we used the Speciteller score for each sentence for both Games and Speciteller. For Games, Speciteller scores tended to skew quite low. Sentences with particularly low specificity scores were often quite short, while sentences with higher specificity scores often contained descriptions of objects used in the games and often contained repeated nouns. For example, the sentence "is that it" is assigned a score of 1e-5 and the sentence "um blue mermaid in the middle and arm in the lower left it's a left arm" is assigned a score of 0.318. For Switchboard, the more specific sentences were more similar to those that appeared in BURNC and often contained many proper nouns.

3.4 Results

3.4.1 Games Phrase Boundaries

Feature Set	Accuracy	F1	Precision	Recall
Best Model	90.5%	0.774	0.890	0.687
Best Model Without Embeddings	90.4%	0.769	0.894	0.676
Best Model Without Positional	90.0%	0.764	0.867	0.686
Best Model Without Syntax	90.1%	0.761	0.884	0.670
Positional Features Only	88.3%	0.711	0.867	0.603
Majority Class Baseline	76.3%			

Table 3.2: Performance scores for **phrase boundary** detection on Games

The best model for phrase boundary detection on the Games Corpus used all features presented in this chapter except for supertags. The performance of this model is presented in table 3.2, along with results when heavily weighted feature groups are removed. (A fuller set of results, including the impact of each feature group in the best model, as well as a list of the 20 most heavily weighted features is included in Appendix B.) We show two baselines, one a majority class baseline (which always predicts no break) and one that uses only positional features. Our best model achieves an accuracy of 90.5% and F1 of 0.774, which is much higher than both baselines, indicating that our model successfully detects phrase boundaries using a variety of features, primarily syntactic and word embedding features, in addition to the positional features in the baseline. This also slightly outperforms the model presented by Rosenberg et al. [24], which only achieves an accuracy of 89.4% and F1 of 0.766. (However, since that model was trained only on the Objects portion of games and also used a 50-50 train-test split, as opposed to our cross-validation approach, it is hard to know how comparable these results are.) It is also worth noting that the accuracy is only roughly 3% worse than our best model on BURNC, despite the fact that Games has no punctuation and slightly less accurate syntactic features.

It is also worth noting that our phrase model has much higher precision than recall. This is likely due to the fact that, throughout the Games corpus, speakers often pause (presumably to think) or end their turns in seemingly unusual places, thereby creating a phrase boundary that our model cannot predict. For example, there are a number of boundaries occurring immediately after determiners. It is also worth noting that while our best model improves considerably on the positional features baseline in terms of both precision and recall, the increase in recall is much higher than the increase in recall. This is unsurprising, due to the fact that the baseline model can only identify phrase boundaries at the ends of sentences or phrase boundaries. These results indicate that our model can find many, though not all, mid-sentence phrase boundaries.

Our results also indicate that some feature groups primarily increase precision, while others primarily increase recall. In particular, removing positional features from the model actually increases recall, though it considerably decreases precision, as well as F1 and accuracy. This is unsurprising, as the model without positional features is much more likely to predict mid-sentence breaks, some (but not all of which) are in the test data. In contrast, removing embedding features, which slightly decreases accuracy, slightly increases precision but decreases recall, indicating that the best model uses word embeddings to identify what types of words that are commonly followed by phrase boundaries (even when they are not at the end of a sentence or otherwise in a position where syntactic features would allow us to predict the boundary).

The bulk of the heavily weighted features in the best model are either positional or syntactic features. The most heavily weighted features are the number of words between the current word and the end of its sentence, and the third most heavily weighted feature is number of words between the current word and the end of its turn. (This is unsurprising, as words at the end of sentences and turns are generally followed by phrase breaks.) Among the remainder of the ten most heavily weighted features, three more are positional features (backwards position in task and the lengths of the current sentence and turn), and four are syntactic features: whether the minimal spanning tree between current and next words is the whole tree (which was the second most heavily weighted feature overall), the width of that minimal spanning tree, the current word's position in its smallest constituent, and the width of the entire syntax tree. The highest weighted feature that is neither positional nor syntactic is the next word's GLoVe embedding (pre-trained on Twitter data), which is the sixth most heavily weighted feature. This indicates that there are common patterns in the semantic properties of words following phrase boundaries. Otherwise, the only feature within the top 20 most important features that is neither syntactic nor positional is the Speciteller score, which is the third most heavily weighted. This is somewhat consistent with BURNC, where Speciteller was the second most heavily weighted feature. The lower position in the Games model is likely due both to the heavy use of positional features in this model and the fact that there are fewer highly specific sentences in Games.

3.4.2 Games Pitch Accents

The best model for pitch accent detection on Games used all features except the domain-adapted embeddings. The performance of this model is presented in table 3.3, along with the results when the best performing feature groups are removed. (As with the phrase boundary model, a fuller set of results is presented in Appendix B.) Again two baselines are presented. One is a ma-

Feature Set	**Accuracy**	**F1**	**Precision**	**Recall**
Best Model	82.1%	0.818	0.814	0.827
Best Model Without Neighboring Words' Embeddings	81.8%	0.816	0.811	0.825
Best Model Without Embeddings	81.7%	0.815	0.806	0.829
Best Model Without Syntax	81.7%	0.813	0.816	0.814
Content Words Baseline	75.0%	0.746	0.742	0.750
Majority Class Baseline	51.1%			

Table 3.3: Performance scores for **pitch accent** detection on Games

jority class baseline (which always predicts no accent); the other classifies all function words (as determined by the LIWC function word category) as unaccented and all content words as accented. Our best model achieves an accuracy of 82.1% and F1 of 0.818, which is much stronger than the content words baseline, which only achieves an accuracy of 75.0% on this data. (No prior work exists for pitch accent detection on this corpus, so we compare solely to the baseline.) These results indicate that our model is relatively strong, mostly due to the use of word embeddings, as well as syntactic features.

Unlike with BURNC, the pitch accent model trained on Games has only slightly higher recall than precision. Furthermore, while the BURNC and Games models have similar accuracies (with the Games model performing slightly better), the best model for Games has much better precision than the pitch accent model for BURNC but much lower precision, indicating that it is easier to identify unaccented words in the conversational domain. This discrepancy may be due to the fact that our model does not always properly de-accent words in compound words or multiword names, in which generally only one word should be accented; these names appear much more frequently in BURNC than in games. Additionally, the content words baseline on Games performs worth than the one on BURNC, indicating that the trend that content words are accented and function words are unaccented is less strong in conversational speech. This difference may be due in part to the fact that Games by design has a large number of content words that are previously mentioned and therefore more likely to be unaccented. However, this does not explain why Games has a higher percentage of accented function words.

Again, it appears that some feature groups increase precision, while others increase recall. In particular, removing syntactic features greatly decreases recall but slightly increases precision, indicating that syntactic features are largely used to identify words that should be accented (such as words at the end of syntactic constituents). In contrast, removing embeddings considerably decreases precision but increases recall, indicating that embeddings can aid in identifying words that should not be accented. This is in part explained by the fact that embedding features were used to identify disfluencies and partial words (which are often unaccented), as well as the fact that some very common content words such as *give* (that might be more likely to be unaccented) are clustered together.

The most heavily weighted feature is the LIWC function word category, followed by word embedding features and word length in syllables. The function word category and syllables feature were also the most heavily weighted in our pitch accent model for BURNC, which is unsurprising, as longer words and content words are more likely to be accented in both genres. Other than those two features, the remainder of the top six most important features correspond to different GLoVe Twitter embeddings of the current word. This is somewhat consistent with our results that show that word embedding features as a group contribute heavily to our model's performance. However, while the gender-neutral embeddings were most useful on BURNC, here, they have much lower weight than the GLoVe embeddings. This difference indicates that word embeddings trained on a similar genre are most helpful, at least within the conversational domain. Other heavily weighted features include the Speciteller score, syntactic features such as the word's position in its smallest constituent and the width of the minimal spanning tree with the next word, and positional features related to turn and task. These features, particularly the syntactic features, may indicate that phrasing has a clear effect on accent. Additionally, the fact that positional features related to turn and task (but not features related to position in sentence) indicate that speakers are less likely to accent words later in their turns and later in the conversation. This effect may be due partly to the fact that given words appear more commonly later in the tasks. However, this is likely not the only reason, as co-reference features that more directly capture the impact of givenness were not heavily weighted.

3.4.3 Games Cross-Validation Folds

Fold ID	**Accuracy**	**F1**
101	90.2%	0.753
102	90.3%	0.771
103	92.1%	0.818
104	88.7%	0.761
105	91.7%	0.809
106	91.2%	0.744
107	86.1%	0.718
108	90.2%	0.733
109	89.7%	0.746
110	91.1%	0.794
111	90.5%	0.769
112	91.0%	0.789
Aggregate	90.5%	0.774

Table 3.4: Performance scores for each fold during cross-validation of phrase boundaries.

As cross-validation was used to computer the results shown above, tables 3.4 and 3.5 report the accuracy and F1 for each fold. Each fold is labeled with the speaker ID number of the speaker whose data was used as the test set in that fold. As is shown in the tables, performance varied quite a bit across different speakers. Both models achieved the highest accuracy on speaker 103; the phrase model has an accuracy of 92.1% and F1 of 0.818, while the accent model had an accuracy of 84.6% and F1 of accuracy 0.841. The phrasing model had the lowest accuracy on speaker 107, achieving an accuracy of only 86.1% and F1 of 0.718. (This is the only fold across both models, where accuracy was worse than one of the baselines, though in this case, F1 was still considerably higher than the positional features only baseline.) The accent model had the lowest accuracy on speaker 102, where it had an accuracy of 79.1% and of F1 0.770.

It is worth noting that there was some consistency between both models. Both models performed much worse than average on speaker 107, and both speakers performed best on speaker 103 and considerably above average on speaker 110. These results may indicate that some speak-

Fold ID	**Accuracy**	**F1**
101	84.1%	0.838
102	79.1%	0.770
103	84.6%	0.841
104	82.7%	0.833
105	82.3%	0.819
106	79.7%	0.781
107	79.3%	0.809
108	83.8%	0.846
109	82.1%	0.832
110	84.0%	0.835
111	82.7%	0.821
112	81.6%	0.812
Aggregate	82.1%	0.818

Table 3.5: Performance scores for each fold during cross-validation of pitch accent.

ers speak in more predictable ways (at least according to the features in our models). However, this is not always the case. For example, the phrasing model performs considerably worse than average on speaker 104, but the accent model performs slightly better than average. One phenomenon that seems unique to the phrasing model is that a few folds, such as those for speaker 106 and speaker 108, had accuracy that was similar to the aggregate or better but much lower F1 due to high drops in recall. These drops likely are due to the fact that these speakers were much more likely to pause in unusual places while speaking, making it difficult to predict all the phrase boundaries in their turns.

When the best models were trained using a random 80-20 train-test split instead of using cross-validation, model performance increased but only slightly. In this case, the phrase model's accuracy increased to 90.7% and F1 increased to 0.781. The accent model's accuracy increased to 82.5% and F1 increased only to 0.819. These results indicate that when training and testing on the same conversations, there is overfitting, but the effect is relatively minor. Additionally, it remains unclear whether excluding a test speaker's conversational partners from the training data had any noticeable effect on the model. In general, the effects of entrainment on prosody require further study.

3.4.4 Games with Durational Features

While the majority of our models focus exclusively on features that can be extracted from only text, in some cases, we may be interested in models that also use a few features related to silence or durations within the utterance. (For example, we may use such a model with text-to-speech system where duration models and acoustic models are trained separately.) In these cases, we segment utterances not just into segments but into interpausal units (IPUs), which are segments separated by at least 50 ms. For these models, we included a word's position in its IPU, as well as the IPU's length, as features. We also directly included the length of silence before and after each word as features.

For phrase boundary detection, adding durational features increases accuracy to 93.1% and F1 to 0.848. This model's most heavily weighted feature is the amount of silence after the word, followed by the distance to the end of the IPU. It is unsurprising that these features improve performance considerably, as phrase boundaries are often accompanied by silence. (In fact, it would be highly surprising if adding silence did not increase performance on this task, since silence is generally considered one of the strongest indicators of a phrase boundary.) These results indicate that, when audio (or at least durational features) are available, it is best to use them to model phrasing.

For pitch accent detection performance also increases, with accuracy increasing to 83.1% and F1 increasing to 0.827. The words until end of IPU feature is the fifth most heavily weighted feature, and the total number of words in the IPU is the eighth most heavily weighted feature. While the increase in performance is less dramatic than in the phrase boundary model, this is to be expected, as the durational features we added are more clearly connected to phrasing. However, these results show that durational features are helpful in predicting pitch accent, and particularly they indicate that words in short IPUs and words closer to the end of IPUs are more likely to be accented.

Feature Set	**Accuracy**	**F1**
Best Model	90.1%	0.897
Positional Features Only	88.7%	0.480
Majority Class Baseline	86.4%	

Table 3.6: Performance scores for **phrase boundary** detection on Switchboard

3.4.5 Switchboard Phrase Boundaries

Both models on Switchboard use all features discussed in this chapter, including the new syntactic features. For phrase boundary detection, we achieved an accuracy of 90.1% and an F1 of 0.897, which is a notable improvement over a baseline that only uses positional features, which had an accuracy 88.7% but a very poor F1 of 0.480 mostly due to poor recall. (The seemingly high accuracy of this baseline is likely due to there being relatively few breaks in the corpus, as the majority class baseline also has a high accuracy of 86.4%.) This is much better than the results reported by Rosenberg et al., who reported an accuracy of 85.7% and F1 of 0.427 on Switchboard [24].

As with Games, the bulk of the heavily weighted features for the phrase boundary model are positional or syntactic features, with the most heavily weighted features being the word's backward position in its sentence, the word's relative position in its sentence, and the size of the minimal spanning tree between the current word and the next word. Interestingly, Speciteller was the 9th most heavily weighted feature. Among the remainder of the twenty most heavily weighted features, the only two that are neither syntactic nor positional are the function word category of LIWC and the cluster ID for the current sentence embedding (based on GLoVe embeddings trained on Twitter). The fact that Speciteller and the function word category of LIWC are so heavily weighted here may indicate there is a strong relationship between phrasing and accent on this corpus, as these features are do not seem directly related to phrasing but are strongly related to accent.

Feature Set	Accuracy	F1
Best Model	76.2%	0.758
Content Words Baseline	68.0%	0.571
Majority Class Baseline	61.0%	

Table 3.7: Performance scores for **pitch accent** detection on Switchboard

3.4.6 Switchboard Pitch Accents

For pitch accent detection, we were able to achieve an accuracy of 76.2% and F1 of 0.758, which is much better than a baseline that accented content words, which had an accuracy of 68.0% and F1 of 0.571. However, our results are slightly worse than Brenier et al., who achieved an accuracy of 77.3% As Brenier et al. looked at a broader context window and mostly did not find linguistic features such as animacy and information status helpful, it may be possible to achieve even better results by looking at some linguistic features over a larger context window.

As with all other pitch accent models, the most heavily weighted feature was the function word category in LIWC. This was followed by Speciteller score, another feature that was also weighted heavily in our pitch accent models for BURNC, though less heavily weighted Games. It makes sense that Switchboard is more similar to BURNC in this case, since Games, which was entirely task-oriented, did not have any highly specific sentences, but Speciteller, where speakers were often encouraged to discuss current events and would sometimes discuss very specific events, had a much broader range. The next three most heavily weighted features are all positional features (relative position in sentence, words until end of sentence, and word number in turn), which may again indicate a strong link between phrasing and accent in this corpus. The remainder of the most heavily weighted features include various word embedding features and a small number of syntactic features, which again, is similar to the results we found on the Games Corpus.

3.4.7 Cross Corpus

In order to see how well our models generalized across corpora, we tested our model trained on Games on Switchboard and tested our model trained on Switchboard on Games. As word embedding features are particularly useful for predicting prosody on conversational speech, we included GLoVe embeddings trained on Twitter data in these models by clustering them over the vocabulary of both corpora. Otherwise, these models used all features discussed in this chapter, excluding the few syntactic features that we extracted only for Switchboard. For these experiments, a single speaker's data was held out at the test set for Games. For Switchboard, the same train-test split was used as in the single corpus experiments. The cross-corpus results are presented in tables 3.8 and 3.9.

			Training Set		
			Games	Switchboard	Naive
Test Set	Games	Accuracy	90.5%	84.3%	76.3%
		F1	0.773	0.820	
	Switchboard	Accuracy	87.1%	90.1%	86.4%
		F1	0.867	0.896	

Table 3.8: Accuracy and F1 scores for cross-corpus evaluation of **phrase boundaries** on Games and Switchboard.

			Training Set		
			Games	Switchboard	Naive
Test Set	Games	Accuracy	82.2%	72.6%	51.1%
		F1	0.819	0.726	
	Switchboard	Accuracy	67.8%	76.2%	61.0%
		F1	0.683	0.758	

Table 3.9: Accuracy and F1 scores for cross-corpus evaluation of **pitch accent** on Games and Switchboard.

These results show that performance of our model does decrease when tested on a different corpus, though the size of this decrease varies between the two tasks. In particular, both phrase boundary models perform reasonably well when tested on the other corpus, with the model trained on Switchboard and tested on Games outperforming the model trained on Games and tested on Switchboard. This difference is likely due to a few factors. First, Switchboard is a larger corpus and contains more prosodically varied speech on a number of topics, while Games is entirely task-oriented dialog focused on object-matching tasks. Second, the Switchboard model was trained using gold standard sentence segmentation and syntactic parses, while Games was not, so any inaccuracies in the syntactic parses in Games may lead to a weaker model that cannot generalize as well.

Performance drops much more in the pitch accent task, with the model trained on Games and tested on Switchboard in particular performing quite poorly. This is again likely due to the nature of the two corpora. Because Games contains only task-oriented dialog from tasks designed to elicit certain patterns of pitch accents, the model does not generalize well to the Switchboard data. Conversely, the Switchboard model's performance also drops when tested on Games, though far less drastically, likely due to the more varied prosody present in Switchboard. Overall, these results indicate that prosody is sufficiently different between task-oriented conversational speech and freeform conversational speech that we cannot expect a model trained on one to fully generalize to the other, although it remains unclear whether the differences would be less pronounced when using task-oriented speech discussing tasks that were not designed to elicit certain prosodic patterns.

3.4.8 Cross Genre

In addition to testing our models across different corpora, we also tested them across different genres to determine whether our models would work cross-genre. In order to do this, we tested our model trained on Games on BURNC and Audix and our model trained on BURNC on Games. Because different word embeddings were used for the different corpora, all word embedding features were removed from these models. Additionally, punctuation features (which are not present in Games) and positional features related to position in turn or task (which are not present in the news corpora) were not used for these models.

The results of the cross-genre experiments are presented in tables 3.10 and 3.11. Single corpus results (with word embedding features removed) and the cross-corpus (but not cross-genre) results of training on BURNC and testing on Audix, as well as majority class baselines, are presented for comparison.

			Training Set		
			Games	BURNC	Naive
Test Set	Games	Accuracy	90.1%	84.2%	76.3%
		F1	0.768	0.442	
	BURNC	Accuracy	85.2%	92.6%	82.8%
		F1	0.137	0.783	
	Audix	Accuracy	79.4%	89.8%	78.5%
		F1	0.079	0.732	

Table 3.10: Accuracy and F1 scores for cross-corpus evaluation of **phrase boundaries** on BURNC, Games, and Audix.

			Training Set		
			Games	BURNC	Naive
Test Set	Games	Accuracy	81.8%	70.5%	51.1%
		F1	0.817	0.630	
	BURNC	Accuracy	70.2%	80.9%	52.8%
		F1	0.562	0.776	
	Audix	Accuracy	69.1%	80.2%	50.9%
		F1	0.557	0.766	

Table 3.11: Accuracy and F1 scores for cross-corpus evaluation of **pitch accent** on BURNC, Games, and Audix.

These results show these models do not generalize across genres, with the models trained on Games and tested on news corpora performing worse than the models trained on BURNC when tested on Games (though all cross-genre experiments led to poor results). The poor performance is especially pronounced in the phrase boundary model, where all cross-genre models perform extremely poorly, with the models trained on Games barely outperforming the majority class baseline and having extremely low F1 scores. This is unsurprising, as apart from punctuation (which is not included in our cross-genre models) and positional features (some of which are not included in cross-genre models), the phrase boundary model primarily uses syntactic features. The syntactic structure of utterances is quite different between the two genres, with the news data containing primarily longer sentences with many clauses and the conversational data containing many fragments and short sentences. Therefore, it is difficult to use a model trained on the syntax of one genre to make predictions about the other.

The performance on the pitch accent task is slightly higher. This is likely due to certain patterns, such as longer words and content words being accented, being consistent across genres. However, these models still perform quite poorly, giving better results than the naive majority class baseline, but generally worse results than the content words baseline. This may be because, particularly once word embedding features (which we have shown greatly contribute to performance on pitch accent tasks) have been removed, these models also use syntactic features heavily, leading to many of the same problems that occur with the phrase boundary task. Overall, while the pitch accent models appear to generalize slightly better, these results show that these models do not perform well in cross-genre settings.

Overall, these results indicate that in order to use a prosody prediction approach of the type discussed in this chapter, it is necessary for the domain of the training data and test data to match, as performance is very poor when training and testing on wildly different domains. While the cross-genre models do technically perform better than the naive baselines shown above (particularly in the case of the pitch accent model), these are very weak majority class baselines. Instead, in cases where it is impossible to train a model on the correct domain, it is likely preferable to use simple heuristics (such as accenting all content words) to do prosody assignment.

3.5 Analysis

Overall, the results in this chapter show that using a wide variety of features appears to be even more useful on conversational corpora than on news corpora, as most of the models we trained in this chapter used nearly all of the features we extracted. This may be a result of the fact that many helpful features are either missing (like punctuation) or more difficult to extract accurately (like syntactic parses), meaning that it is more important to have a wide variety of features.

While positional features were not particularly useful when building models on BURNC, they were extremely useful for our models on both Games and Switchboard. This is because there is no punctuation in either conversational corpus, so the model uses positional features to predict phrase breaks between sentences. Additionally, since our sentence segmentation method for Games was not perfectly accurate, the position in turn feature is very useful, since speaker turns are almost always followed by breaks.

Once again, syntactic features are extremely helpful for both corpora for both tasks. Again, this is completely unsurprising for the phrase boundary detection task, as words at the end of syntactic constituents are more likely to be followed by phrase boundaries. Interestingly, while the BURNC phrasing model relied very heavily on syntactic features (and became worse than the punctuation baseline when they were removed), removing syntactic features only decreased accuracy on the Games phrasing model by 0.4%. This may indicate that the Games model in particular relies less on syntactic features due to some inaccurate parses. However, since Switchboard, which also has many fragments and disfluencies but has gold standard parses, used similar syntactic features, this may simply indicate that syntactic features are less crucial because the model makes heavier use of positional features and word embeddings. Interestingly, supertags were not useful on the Games corpus (though they were useful on the Switchboard corpus.) It is unclear why this is the case, though it is possible that, due to the sentence structure of Games, the supertagger (which was trained on news data) incorrectly tagged too many words. For our pitch accent models, syntactic features were also quite helpful. However, since the most heavily weighted syntactic features were largely features heavily associated with phrasing, like the width of the spanning tree between the current word and the next word, which may indicated not that syntax directly affects accents but that phrasing affects accents.

As for word-level features, we once again find that the function word category of LIWC is the most useful feature for pitch accent prediction, reflecting that most function words are unaccented and most content words are accented. Interestingly, this feature was also weighted highly in the Switchboard phrase boundary model. This may indicate that content words are more frequently followed by phrase boundaries or may more indirectly indicate that there is a strong relationship between phrasing and pitch accent. The number of syllables in a word was also a useful feature for pitch accent prediction, though it was more useful on Games than on Switchboard. This may be because Switchboard has a smaller proportion of accented words than the other corpora, so it may have more unaccented long words.

Word embeddings appear to be extremely useful features for both tasks on both corpora, with word embedding features being heavily weighted for both tasks on both corpora. This may be because, even in sentence fragments or other utterances with unusual syntax, word embeddings can provide information about the current word, its neighbors, and its sentence. However, again, it is hard to draw strong conclusions while using our clustering approach, which removes a lot of information that can be gained from word embeddings. In terms of the specific embeddings we looked at, overall the GLoVe embeddings trained on Twitter were useful on both corpora, whereas the domain-adapted embeddings were not. This indicates that, while there are differences between Twitter data and conversational speech, the two domains are similar enough that word embeddings trained on one can be used on the other (at least when using clustering first) and that domain adaptation is unnecessary. We also found that the gender-neutral embeddings were very useful on Games, which once again indicates that removing semantic information that has little effect on prosody can be useful.

Speciteller once again is useful in both models on both corpora and is heavily weighted in pitch accent models on both corpora. This indicates that there are is a relationship between prosody and specificity in the conversational domain, and in particular that more specific sentences have more accented words. Interestingly, the Speciteller feature is very heavily weighted in the Switchboard phrase boundary model but not particularly heavily weighted (though still useful) in the Games phrase boundary model. This discrepancy may be due to the fact that the Speciteller scores on Games have a much smaller range, since the utterances are largely related to gameplay and do not include proper nouns or other highly specific information. Since Speciteller was also useful in the BURNC phrase boundary model, so this may indicate a relationship between specificity and phrasing in both the news and conversational domains. However, since the Switchboard phrase boundary model also weighted the LIWC function word category heavily, the importance of Speciteller may simply indicate a strong relationship between phrasing and accent on this corpus.

Finally, co-reference features were helpful for both tasks on both problems. However, they only improved performance slightly. In particular, while Games was developed to explore prosodic differences between given and new information, co-reference features were not particularly highly weighted in its pitch accent model, indicating that syntactic and semantic information represented by other features may simply have a stronger effect on pitch accent than information status, even in a corpus with a large number of given words. It is also somewhat interesting that co-reference slightly improved both phrase boundary models, even though co-reference does not appear to have any relationship to phrasing. However, this may simply be related to the fact that pitch accent appears to affect phrasing, and co-reference affects pitch accent. Given the weak effect found, it is worth looking further into the effects of information status on prosody, since it is widely believed that information status has a strong effect on pitch accent, particularly within the conversational domain. The results presented here may show that this effect, while existent, is weaker than believed, but it is equally possible that the effect would become clearer when explored in isolation and not alongside other linguistic features, or in experiments where a more robust set of co-reference features (such as those that find synonyms) are used.

3.6 Conclusions

In this chapter, I presented models for predicting prosodic events on two conversational corpora: the Columbia Games Corpus and the Switchboard Corpus. Despite the lack of punctuation in these corpora, we were able to achieve strong phrase boundary detection models with accuracies over 90% by using positional features, syntactic features, word embedding features, as well as a large number of other linguistic features. Additionally, we were able to achieve strong pitch accent detection models, particularly on Games, where the best pitch accent model slightly outperformed the best pitch accent model on BURNC. Interestingly, while the Games Corpus was developed to explore the relationship between information status and prosody, co-reference features were helpful but had a relatively minor impact on the model.

We also showed that, while these models are strong, they do not generalize well across genres. Even when training on Games, which is task-oriented, and testing on Switchboard, which is open-ended, performance decreased. Even more notable, training on Games and testing on BURNC or training on BURNC and testing on Games produced extremely poor results. These results indicate that these models can only be used on test data that is relatively similar to the training set. These results also emphasize the importance of modeling prosody specifically using conversational data when attempting to synthesize conversational speech, as it is substantially different from the prosody of read speech.

While the results presented in this chapter are very promising, there is still room for improvement on all models discussed in this chapter. Since word embeddings appear to be extremely useful for prosody modeling on conversational speech, one approach that is worth exploring neural models that can more fully make use of them. Additionally, given the low recall of the phrase boundary model, it may be worth further investigating the phrase structure of conversational corpora and determining if other features could help predict mid-turn and mid-sentence phrase boundaries. (It may also be worth looking into whether it is desirable to reproduce all the phrase boundaries present in natural conversational speech, as some correspond to a speaker trailing off or pausing to think, which may not be desirable behaviors for a voice assistant or other synthesized voice.) Finally, for ToBI label prediction in general, it is worth examining semi-supervised approaches. While Games in particular has been completely ToBI annotated through manual and automatic methods, it has under 10 hours of data, which is a fairly small amount of data by modern standards, and the other ToBI-labeled corpora discussed in this chapter and the previous chapter have even less labeled data.

There is particularly room for improvement on the pitch accent model on Switchboard. In addition to the approaches discussed above, Switchboard has gold standard annotations for some features, such as contrast and animacy, which we did not use and that may improve the pitch accent model. Finally, the Switchboard Dialog Act Corpus provides dialog act tags for about half of the Switchboard Corpus [57]. We did try to include dialog act tags directly as features to our prosody model and found that they did not improve performance. However, there is a clear relationship between dialog acts and prosody of conversational speech, and it may be possible to develop a prosody model that incorporates information about dialog acts (although this prosody model may need to predict something like sentence contours instead of simply predicting the locations of pitch accents and phrase boundaries).

Chapter 4: Generating Prosodically Appropriate Speech Using Merlin

4.1 Introduction

In the previous chapters, I have presented models that can predict phrase boundaries and pitch accents from text. Now, I will discuss work that uses these models to synthesize speech with appropriate prosody. While there is a large body of work on improving the prosody of synthesized speech, very little of it directly uses ToBI labels, instead learning prosody through seeing large amounts of data or through features that are less directly connected to the actual prosodic output. Many of these models do output high quality prosodically appropriate speech, but because of the nature of the prosody model, it can be hard to adjust the prosody of an utterance. In contrast, many commercial TTS systems do allow a user to specify the level of emphasis on certain words. However, when no user input is given, these systems use default prosody models that may or may not be accurate.

The model presented in this chapter instead directly incorporates the phrase boundaries and pitch accents predicted by the models presented in previous chapters into the TTS pipeline. Because our prosody prediction models are very accurate, this model will almost always produce appropriate breaks and accents without any further user intervention. However, because breaks and accents are generated at an earlier stage than the actual synthesis, it is easy to adjust the prosody of an utterance.

In this chapter, I will present a pipeline for producing prosodically appropriate speech using the Merlin toolkit for TTS. This model was presented at Speech Prosody 2022 [2]. We use Merlin because, unlike end-to-end TTS systems, it extracts features from text before training, allowing us to easily add new prosodic features. Additionally, unlike end-to-end systems, Merlin can be trained on a relatively small amount of data, so we were able to train it on a few hours of data from

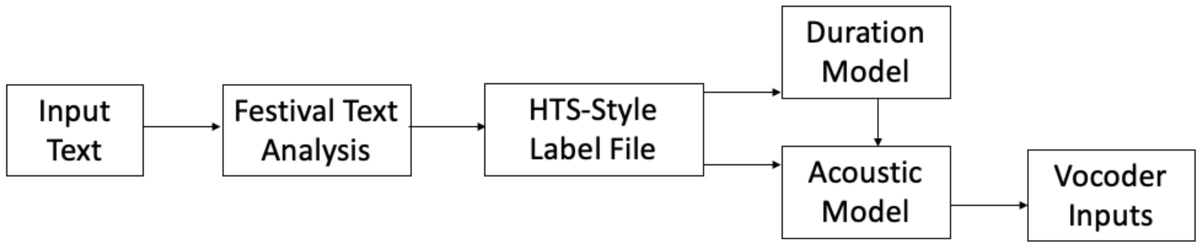

Figure 4.1: Diagram representing the default Merlin pipeline

BURNC.

4.2 Merlin Toolkit

Merlin is an open source toolkit for neural network speech synthesis [32]. It was released in 2016 by researchers at the University of Edinburgh. Unlike more recent end-to-end systems, it contains several components. First, text is analyzed linguistically, and a large number of features are extracted for each phone in the input text. These features are used as input to a a duration model, which uses a deep neural network (DNN) to determine the length of each phone. Then, a similar DNN is used to predict acoustic parameters, specifically Mel-Ceptstral Coefficients, band aperiodicities, and $f0$, for the utterance. Merlin's default neural network architecture, consisting of 6 hidden feed-forward layers of size 1024, is used for both the duration and acoustic models for all experiments. Finally, the acoustic parameters are passed to a vocoder to create audio. We used the open source WORLD vocoder for this purpose. A full diagram of the default Merlin pipeline is presented in figure 4.1.

The input to the duration and acoustic models takes the form of label files where each phone is listed along with about 50 features. The label files contain the same features and are in the same format as those used as input to HMM-Based Speech Synthesis (HTS) toolkit, an older HMM-based parametric speech synthesis toolkit [58].The bulk of these features include simple features related to the identity of the current and surrounding phones, as well as a large number of positional features, related to the phone's position in its syllable, word, phrase, and utterance. Other features include part of speech for current and surrounding words, features related to stressed syllables, and the vowel in the current syllable. All of these features are extracted using the Festival toolkit [59], another older text-to-speech toolkit, which performs basic linguistic analysis, including pronunciation modeling and part of speech tagging. It also uses ergodic HMM alignment to align audio with transcripts at the phone level.

Finally, Merlin converts the features within the HTS label file into slightly under 500 features using a questions file that lists all Merlin input features. Primarily, this file normalizes the inputs so that all features have values between 0 and 1. The majority of the changes come from converting some features in the label files into binary features. For example, the phone identity feature in the HTS label file becomes nearly 50 binary features corresponding to different phone identities. Otherwise, all added features correspond to phonetic properties of the current phone or the the current syllable, such as whether the current phone is a voiced stop or whether the current syllable contains a rounded vowel.

Among the features included in the Merlin front end, there are a small number of features related to prosody. Many of these features are related to phrasing. These include features related to the position of the current syllable in the current phrase, the position of the current word in the current phrase, and the position of the current phrase in the utterance. Other features include the number of syllables and words in each of the current, previous, and next phrases. By default, phrase boundaries are determined by Festival using a very simple decision tree model, which takes into account only punctuation and part of speech as features. Additionally, the HTS label files often include pauses after phrase boundaries (as predicted by the decision tree model), which, due to the simplistic nature of the model, can often lead to infelicitous pauses, particularly after punctuation that does not indicate phrase boundaries. (For example, the label file for a sentence mentioning *"Boston, Masschussetts"* would often contain a pause between *Boston* and *Massachussetts*.)

In terms of accent, the HTS label files include binary features for whether the current, previous, and next syllables are accented, as well as features related to the distance between the current syllable and the closest accented syllables. In order to determine which syllables are accented, when generating label files, Festival uses a simple heuristic, treating any stressed syllable in a content word as accented and all other syllables as unaccented. Despite its simplicity, this is a fairly accurate model. In fact, on BURNC, using only function words (as determined by LIWC) to predict accents results in an accuracy of 79.9%, which is worse but not drastically worse than the 81.9% accuracy of the best model presented earlier. Therefore, while our model makes some minor improvements by including more accurate accent labels, more substantial improvements come from improving phrasing features. Finally, HTS label files contain one feature for the ToBI endtone of the current phrase. However, in Merlin, altering this feature has little effect on the output audio, and by default, Merlin does not use it (and therefore we do not use it in any experiments in this chapter.)

4.3 Approach

In order to train Merlin to produce phrase boundaries and pitch accents in the predicted locations, we update the front end to account for them, while leaving the rest of the system unchanged. First, we removed the existing phrasing model. This is primarily because of the problem mentioned in the prior section where the prosody model would insert infelicitous breaks. We also found that simply altering the existing phrasing features had little effect on the synthesized audio. Therefore, instead of altering those features, we added an additional feature to indicate whether the current word came directly before a phrase boundary. While we only consider 4-level phrase boundaries, we found the need to make a distinction between breaks that appeared mid-sentence and those at the end of a sentence, as they had noticeably different acoustic properties. (This may be in part due to the fact that pauses are often longer after sentence breaks than after mid-sentence breaks, and the duration model would treat a pause after a word as part of the previous phoneme, as we removed explicitly encoded pauses from the label files.) For accents, we simply added a binary feature for whether the current word was accented or not.

In order to create these label files, we used the best models presented in chapter 2 trained on BURNC for predicting phrase boundaries and pitch accents. We ran these models on all utterances

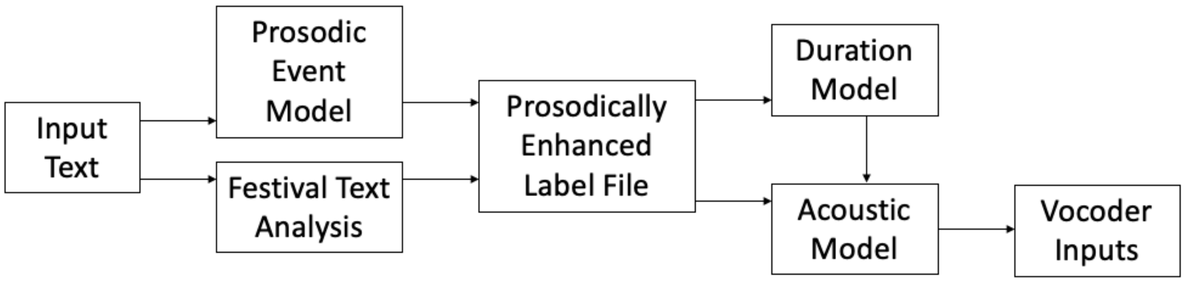

Figure 4.2: Diagram representing the prosodically enhanced Merlin pipeline

in BURNC from the three female speakers. (As we did no speaker conditioning, and the final voice was effectively an average of the three training voices, we could not train on the data from both male and female speakers.) For the sake of consistency, only predicted prosodic events were used when creating prosodically enhanced label files, even for utterances where gold standard labels were available. (However, since some utterances were in the training set for the prosody model, it is likely that the predicted events were more accurate on these utterances than on those that were never manually labeled.) The output of these models was then added to standard HTS label files generated using the standard Festival toolkit with only the phrasing model removed, resulting in new prosodically enhanced label files. The questions file that converts between the HTS label files and the input to Merlin's models was updated to include our new features, including reducing our new trinary phrase boundary feature into two binary features, as Merlin reduces all categorical features to binary features. Otherwise, Merlin's standard pipeline, using DNNs for both duration and acoustic models, was run on the data. This modified pipeline is displayed in figure 4.2.

The data set used to train Merlin consisted of 725 utterances, of which 60 random utterances were held out as a validation set, another 60 held out as a test set, and the remaining 605 used as the training set. In total, this added to about three and a half hours of data, with about 50 minutes of data from each of speakers f1a and f2b and the remainder (about 110 minutes, or slightly under half of the data set) from speaker f3a. Each utterance consisted of a short paragraph from a radio news story. Utterances ranged in length from one to about four sentences, and averaged about

Table 4.1: Comparison of MCD and RMSE of f0 between our prosodically enhanced pipeline and the Merlin baseline

Model	**MCD (dB)**	**RMSE (Hz)**
Our pipeline	5.014	44.586
Baseline	5.053	45.016

16 seconds in length, though average length varied from speaker to speaker. Speaker f2b, whose portion of the corpus included longer and more in depth stories, averaged over 20 seconds per utterance, and speaker f1a, who mostly read shorter updates, averaged slightly over 11 seconds, with speaker f3a tending closer to the overall average. In order to test how our pipeline worked on longer utterances with many phrases, we did not segment any utterances beyond the paragraph level.

4.4 Results

4.4.1 Objective Results

First, we compared the performance of our test set to the original BURNC audio. We calculated mel-cepstral distortion (MCD), which measures the difference in mel cepstra between two sequences, as well as root mean squared error of $f0$ between audio synthesized with our pipeline and the original BURNC audio, as well as the same measures between audio synthesized with the baseline Merlin pipeline and the original BURNC audio. Dynamic time warping was used when calculating these numbers in order to account for typical differences in timing. The results are presented in table 5.2. MCD is a commonly used metric to determine the similarity of synthesized speech to natural speech. While it does not specifically capture improvements to prosody, it provides an overall measure of voice quality. Additionally, we calculate the RMSE of $f0$ to have a metric that better capture improvements specifically to prosody.

These results indicate that our model performs better on both metrics, though the differences are quite small. The small size of the difference may be due in part to the fact that both synthesized voices are trained on data from multiple speakers and therefore have different acoustic properties from any of the individual speakers in the training data. However, it also may indicate that there are still voice quality and naturalness issues remaining with our pipeline. Still, as our pipeline did perform better in both metrics, this indicates that it produces more natural humanlike speech than the baseline.

4.4.2 Listening Test

In order to test how this model worked on novel sentences within the radio news genre, as well as to test whether our pipeline truly produced more natural output, we also tested the same Merlin model, trained on the same data, on paragraphs from recent news stories. Specifically, we found 20 paragraphs from a variety of news stories presented on *Morning Edition*, a news program on National Public Radio (NPR), in 2020. For consistency with the BURNC data, all paragraphs ranged in length from one to four sentences. (Specifically, since we were concerned with how Merlin would produce longer utterances with complicated phrasing, only two of the utterances were one sentence long, and in both cases, they were sentences with multiple clauses.) While these paragraphs were all taken from transcripts of stories that aired on the radio, we used only the transcripts and not the audio, particularly because the variety of newscasters and the presence of music or background noise (from stories reported on location) in many of the stories made the audio difficult to compare to either the original BURNC audio or the synthesized audio.

The 20 utterances were synthesized both using the standard Merlin pipeline and our pipeline presented in the prior section. As with the BURNC data, we determined the locations of phrase boundaries and pitch accents for the prosodically enhanced label files using the best model trained on BURNC. Then, the synthesized utterances were presented to workers on Amazon Mechanical Turk. For each utterance, each worker was presented with two versions of the utterance side by side, one produced with each pipeline, and had to select which one they believed were more natural. Workers were presented with one pair at a time and were required to listen to the entirety of both versions before submitting their choices. This was a forced choice task, so there was no neutral or “no preference” option. The order in which the two versions of the utterances were presented was randomized so that for 10 utterances, the baseline was presented first, and for the other 10, the baseline was presented second. The order in which the different utterances was presented was also randomized. Finally, to ensure that workers were completing the task properly, three check questions were interspersed throughout the task. For these questions, workers were presented with one intelligible utterance synthesized using Merlin and one clip of garbled speech, which was output from a Merlin experiment where certain features were configured improperly. All workers answered these questions correctly and selected the intelligible speech.

5 workers judged each utterance, for a total of 100 judgments. Of these, 80% preferred the utterances synthesized using our pipeline with the prosodically enhanced label files over the baseline. This was a significant preference with a p-value of $1.97 * 10^{-9}$. This indicates that adding prosodic event features directly to the front end of Merlin creates substantial improvements to naturalness, particularly when synthesizing longer utterances.

4.5 Conclusion

In this chapter, I have presented a modified Merlin pipeline that greatly improves naturalness on longer utterances. This pipeline uses the model discussed in chapter 2 to predict the locations of phrase boundaries and pitch accents within all training and test utterances. These prosodic events are then directly incorporated as input features when training Merlin. This approach produced much more natural speech than the baseline Merlin pipeline, particularly when synthesizing longer utterances, as listeners overwhelmingly preferred speech synthesized using the modified pipeline.

While the work in this chapter serves as a strong proof of concept to show that TTS can be improved by directly incorporating the locations of prosodic events, Merlin still struggles with overall voice quality. In particular, Merlin (like many TTS systems) particularly struggles with longer utterances in the baseline condition. Since our experiments focused on longer utterances, this made the improvements fairly stark. Therefore, it is important to perform similar experiments on higher quality systems (such as end-to-end systems), which provide a stronger baseline. In the next chapter, I will present preliminary work that modifies an end-to-end TTS system to incorporate prosodic events.

Chapter 5: Incorporating Prosody into End-to-End Text-to-Speech

5.1 Introduction

In the previous chapter, I showed that we were able to improve the output of a TTS system trained using Merlin by incorporating prosodic features. In this chapter, I present similar work using an end-to-end TTS pipeline.

End-to-end approaches to speech synthesis have become common in the past five years. They generally improve voice quality over other neural TTS systems (such as Merlin) by operating autoregressively, using neural text encoders, and using attention to align text and speech [60]. However, it can often be difficult to control output in end-to-end systems, as they involve no front end or back end feature extraction and instead take in text (or, in some cases, phones) and output spectrograms. Additionally, end-to-end models require a larger amount of data than other TTS systems, making it impossible to train on BURNC, which only contains a few hours of data.

In this chapter, I present work using the end-to-end TTS system DCTTS. I show how incorporating phrase boundaries into the input to DCTTS improved a TTS model trained on LJSpeech, a commonly used corpus of read speech. I present preliminary results from this experiment, showing objective results on news data as well as the results of a listening test. I also discuss possible future work for incorporating prosody into an end-to-end pipeline.

5.2 End-to-End TTS

For the past half decade, end-to-end TTS has been considered the standard approach. Unlike other parametric TTS systems, these systems require little pre-processing or post-processing. In particular, the front end of an end-to-end to end system simply takes in the sequence of characters from the text that is to be synthesized. While it is common to perform text normalization to remove numbers, symbols, and abbreviations from the text, generally no linguistic features of any sort are extracted from the text or included as input to the model. (In some case, pronunciation modeling is also done prior to training, and a string of phones is used as input instead of a string of characters. In these cases, there are still no other linguistic features included.) Similarly, instead of outputting acoustic features, end-to-end system generally output spectrograms.

The first and best known end-to-end system is Tacotron [33], introduced in early 2017 by researchers at Google. It is an RNN-based system, which uses an attention-based encoder-decoder model. In particular, it processes the front end (effectively substituting for manual feature extraction) by passing the string of characters through a series of convolutional layers and highway layers. The outputs of these layers are then passed to an RNN-based encoder, which is connected to an attention module used to align the text and the audio. Similar recurrent architectures exist for encoding and decoding audio, allowing the system to work directly with (and output) spectrograms, which can be converted into waveforms either using a neural vocoder or the Griffin-Lim algorithm [61].

The current state of the art is Tacotron2 [34], which uses an architecture very similar to the original Tacotron but using Wavenet [35], a neural system for outputting high quality waveforms, on the back end. When trained on a very large clean training set, Tacotron2 can create incredibly high quality speech, which in some cases, human listeners preferred to natural human speech. However, this system takes a long time to train, making it difficult to use in research applications. In particular, for research that does not focus on the back end of the system, it is very common to use the Griffin-Lim algorithm to produce waveforms instead of using Wavenet or another neural vocoder. Additionally, since Tactotron2's output is very high quality, but the training process is quite slow, current work on end-to-end systems that substantially vary significantly from Tacotron focuses largely on improving efficiency. These approaches often remove the recurrent layers from Tacotron and sometimes remove the autoregressive aspects of the pipeline altogether. For example, DCTTS [62], discussed further in the next section, uses primarily convolutional layers, and Fastspeech [63] uses a Transformer-inspired feed-forward network. (Unlike for many NLP problems,

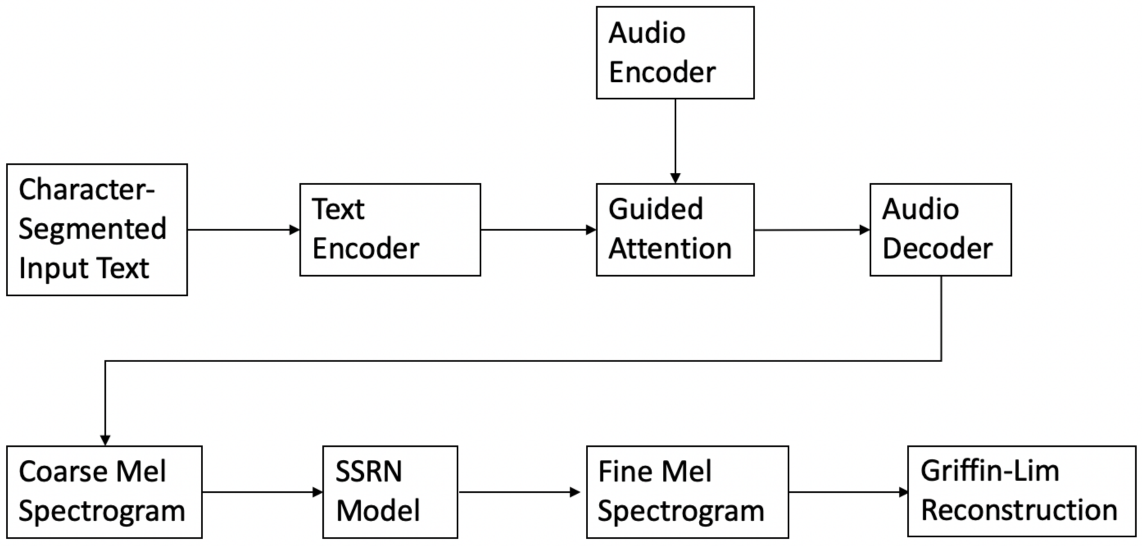

Figure 5.1: Diagram displaying the entirety of the standard DCTTS pipeline

autoregressive Transformers are not commonly used for TTS currently.)

5.3 DCTTS

One relatively efficient pipeline for end-to-end TTS is Deep Convolutional TTS (DCTTS). DCTTS was presented by Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara in 2018 [62]. The goal of DCTTS was to create a more efficient pipeline that can capture the improvements of Tacotron by using and end-to-end pipeline that uses attention as well as neural text and audio encoders. This is done by largely using convolutional layers to process both text and speech, as opposed to the recurrent layers used in systems like Tacotron, which greatly decreasess both training and inference time. In their original paper, Tachibana et al. found that listeners rated speech synthesized using a DCTTS model that had been trained for 15 hours comparably to speech synthesized using an open source version of Tacotron trained for 12 days.

Figure 5.1 displays the pipeline used to train DCTTS. DCTTS breaks training down into two two stages. The first model, the text2mel model, is represented in the top half of the figure. The text2mel model takes in the plain text for an utterance and outputs a coarse mel spectrogram. By default, this model uses only text; no pronunciation modeling or other linguistic feature extraction is done prior to training. The text encoder, audio encoder, and audio decoder used with the text2mel model are all fully convolutional, and a guided attention model. The second row of the figure demonstrates how the coarse spectrogram is converted into a waveform. This starts with the second model, the spectrogram super-resolution network (SSRN), which takes in the coarse spectrogram and produces a full spectrogram. The SSRN model is fully convolutional. The spectrogram it outputs can then be converted to a waveform using either a neural vocoder or the Griffin-Lim algorithm.

Various implementations of DCTTS are widely available. For all experiments discussed in this chapter, we used a publicly available Pytorch implementation [64], making modifications as needed to the front end incorporate prosodic features. Other than changing the front end, we did not change the DCTTS pipeline in any way. All waveforms were produced using the Griffin-Lim algorithm.

5.4 Corpus

Because our BURNC training set contains under four hours of speech, training DCTTS directly on BURNC resulted in very poor quality audio. Instead, for all DCTTS experiments we used LJSpeech, a larger corpus of read speech.

The LJSpeech Corpus [65] is a public domain corpus of read speech. It contains approximately 24 hours of data taken from 7 nonfiction audiobooks, all read by the same female speaker. The speech is taken from LibriVox, a project that creates a large number of free public domain audiobooks. The corpus is segmented at the sentence level, and clips range between roughly 1 second and 10 seconds in length, with a mean length of 6.57 seconds and 17.23 words. As this corpus is large, clean, and publicly available, it is very commonly used in TTS experiments.

The corpus is fully transcribed but has no other annotations. When training TTS systems on this corpus, we use a normalized version of the transcript (provided with the original corpus), which expands out numbers, symbols such as dollar signs, and common abbreviations. All other features used in our modified pipeline were extracted using Stanford CoreNLP [45], AuToBI [4], or other NLP tools discussed in chapter 2.

5.5 Methods

In order to add phrase boundaries to the DCTTS pipeline, we adjusted only the input. By default, our DCTTS implementation converts the text to all lowercase and removes all characters are not in its vocabulary, which contains all letters, spaces, and a small number of punctuation marks. (Since the text was normalized prior to training, most numbers and symbols had already been removed.) In order to include phrase boundaries, we simply add a new character to the vocabulary, corresponding to the location of a phrase boundary. This character is inserted immediately after the last character of the word that precedes the boundary. (Crucially, there is no space between the word and the break symbol, though a space is inserted after the break symbol.) Since all training and test utterances were only one sentence long, we did not distinguish between mid-sentence breaks and breaks at the end of the sentence. (Furthermore, it is unlikely that it would be necessary to make this distinction even in longer utterances, since end of sentence punctuation is already a part of the input text.)

Since our text-based prosody prediction models are highly domain-sensitive, and we did not train DCTTS on news data, we instead chose to use AuToBI to find the location of phrase boundaries. AuToBI is a tool that predicts ToBI labels based only on audio [4]. Since the goal during training was for the model to learn the acoustic cues associated with phrase boundaries (in order to properly synthesize them), we found that it was more useful to use audio features to find the boundaries in the training set. To find these boundaries, we first used the toolkit Aeneas to get word-level alignments [66], then used an AuToBI model trained on BURNC.

5.6 Results

5.6.1 Objective Results on LJSpeech

First, to test our DCTTS model, we held out a set of 20 sentences from LJSpeech and synthesized them using DCTTS. For these test utterances, we determined the placement of phrase boundaries using AuToBI. We used the phrase boundaries from AuToBI for two reasons. First, this allowed us to test the effects of training using phrasing information in isolation without using a text-based prosody model. Effectively, this allows us to see the "upper bound" of our current approach by seeing the effects of incorporating a perfect prosody prediction model into the pipeline. Second, due to differences in genre between LJSpeech and BURNC, as well as the non-deterministic nature of prosody, our text-based model often predicted phrase boundaries that were not present in the original audio. This posed challenges when evaluating the test utterances, since most objective metrics involve comparing synthesized audio to the original natural audio.

Table 5.1: MCD scores on a held out LJspeech test set for two DCTTS models: one trained with phrase breaks included in the input and one trained with the default pipeline

Model	**MCD (dB)**
With Breaks	8.091
Baseline	8.266

We calculated mel-cepstral distortion (MCD) between utterances from the original LJSpeech corpus and utterances synthesized using DCTTS with phrase boundaries included in the input. We compared these results to the MCD between the original and utterances synthesized using a baseline where DCTTS was trained on LJSpeech with no alterations to the input (or any other aspect of DCTTS). While MCD does not particularly capture aspects of prosody, but it serves as an overall comparison between synthesized speech and natural speech and is commonly used to determine overall voice quality.

The results are presented in 5.1. These results clearly show that adding breaks during training, reduces MCD, indicating that our pipeline creates utterances that are more similar to the original

LJSpeech audio. However, MCD is somewhat high for both models, indicating that the current implementation of DCTTS may output some lower quality utterances in both the baseline and our model.

5.6.2 Objective Results on News Data

As with Merlin, we wanted to test our full pipeline on modern news data. Again, we used a set of test utterances taken from NPR's *Morning Edition*. As our model was trained only on single sentences and not longer utterances, it was unable to produce satisfactory output for longer utterances. Therefore, we segmented utterances at the sentence level. The placement of phrase boundaries for these sentences was determined base only on text-based features extracted using NLP tools, using the best model presented in chapter 2.

For the results presented here, we used 12 test sentences taken from 5 separate news stories. Each of the news stories was read by a different broadcaster. The original audio for each of these sentences ranged in length from 1 second to 13 seconds. Because we wanted to directly compare the original NPR audio to our synthesized audio, we selected only sentences that were read by female speakers and had no additional background noise or music.

Table 5.2: MCD scores on NPR for two DCTTS models: one trained with phrase breaks included in the input and one trained with the default pipeline

Model	**MCD (dB)**
With Breaks	12.738
Baseline	12.855

We calculated MCD between utterances from the original NPR broadcasts and utterances synthesized using DCTTS with phrase boundaries included in the input. Again, we compared these results to the MCD between the NPR utterances and utterances synthesized using a baseline where DCTTS was trained on LJSpeech with no alterations to the input. The results are presented in table 5.2. While MCD is fairly high for both models, this is mostly likely due to the fact that DCTTS models were not trained on any speech from the speakers in the test set (or, in fact, from any radio presenters) and is not necessarily indicative of particular voice quality issues in the NPR utterances. Additionally, we can see that MCD is slightly lower for the model that uses breaks than for the baseline model. This is a promising sign, as it indicates that adding breaks produces speech that better replicates radio broadcast speech.

5.6.3 Listening Test

The same 12 sentences used to calculate MCD were also used as a test set for a subjective listening test. As with the listening test we used for Merlin, listeners were presented with two versions of each utterance: one synthesized using a model trained with phrase boundary information, the other synthesized using the DCTTS baseline model. The order in which the two utterances were presented was randomized so that for half of the sentences, listeners heard the baseline first. For each pair of utterances, each listener was prompted to pick which of the two voices sounded more natural. Listeners were required to choose one of the voices; there was no neutral or "no preference" option.

As with Merlin, three check questions were interspersed into the task. Each of the check questions had the listeners select between one intelligible utterance (of varying quality, synthesized as part of other Merlin or DCTTS experiments) and one highly unintelligible utterance (created by a misconfigured TTS system). Users who did not answer all three check questions correctly were removed from the data. After users who incorrectly answered check questions or did not properly complete the task were removed, the final data includes results from 8 listeners, all of whom listened to all 12 test utterances.

Out of the 96 judgments included in this test, 58.3% preferred the model that incorporated phrase boundaries. This result indicates that listeners have a slight preference for our model. However, it is not statistically significant, with a p-value of 0.102. These results make it hard to draw strong conclusions.

Table 5.3 presents the percentage of listeners who preferred the voice with phrase breaks over the baseline. The sentence IDs refer to the order in which listeners heard the sentences, which

Sentence ID	Preference for System with Breaks
1	50.0%
2	62.5%
3	50.0%
4	75.0%
5	25.0%
6	50.0%
7	75.0%
8	100.%
9	12.5%
10	50.0%
11	37.5%
12	100.%

Table 5.3: Percentage of listeners who preferred the experimental voice (where phrase boundaries were incorporated) to the baseline system for each individual utterance presented in the listening test.

was randomized but consistent for all listeners. The ordering has no relation to the order in which sentences appeared in the original news stories.

As is shown in the table, for about half the utterances, there was no strong consensus between listeners, with four utterances where exactly half of the listeners selected the baseline. However, there where four utterances where at least six of the eight listeners preferred the model with breaks, including two utterances where all eight listeners preferred the version from the model with breaks. Interestingly, one of those utterances was also the longest utterance in the test set, indicating that adding breaks can improve quality on longer utterances but may have no impact (or even worsen voice quality) on shorter utterances. However, overall, there were a number of low quality productions and other audio artifacts in both the baseline utterances and those using phrase boundaries. While the audio artifacts were in different places across the two models, some of the stronger listener preferences (for the baseline or our model) may reflect that there were fewer intrusive audio artifacts, not that one voice had better phrasing or prosody.

5.7 Future Work

While our preliminary results on incorporating phrase boundaries into DCTTS are promising, they are also fairly inconclusive, and there is still more work to do in developing a full prosodically enhanced end-to-end pipeline. In particular, the listening test results seem to indicate that adding phrasing to the model made a slight improvement, but the results are not statistically significant. As many of the decisions on the listening test seem to be due to overall issues with voice quality and audio artifacts, it may be worth exploring different ways to configure DCTTS (or even using a slightly different end-to-end model or training set) to see if phrase boundaries continue to improve the model when working with a higher quality model.

Another obvious next step is to incorporate pitch accent into this pipeline. While with Merlin, most of the improvements to voice quality came from incorporating phrase boundaries, this is largely due to the fact that the Merlin baseline used a strong heuristic (accenting content words) to predict accent. However, in DCTTS there is no feature extraction stage, and the baseline model does not always place accents appropriately. However, using our current approach, it is more difficult to incorporate accents, as we cannot simply insert accents into the input the same way we can insert breaks. Currently, we define a new set of characters in our vocabulary corresponding to letters in accented words. However, preliminary attempts to use this approach led to worse voice quality. This may indicate that we need to adjust hyperparameters to account for the additional input features or may indicate we need to try another approach.

In the longer term, we hope to train DCTTS using prosodic features on longer utterances. When using Merlin, we found that using our updated pipeline had a stronger impact when synthesizing longer (paragraph-length) utterances. This impact comes from the fact that necessary phrase boundaries were often missing in longer utterances trained with the baseline system, leading to unusual prosody over the longer utterance. Additionally, producing appropriate phrasing and accents is particularly important when synthesizing longer utterances, as inappropriate prosody can make it very difficult for a listener to follow longer audio.

So far, we have been unable to produce longer utterances using DCTTS. This is largely due to the fact our training data only contained single-sentence utterances and contained no utterances longer than 10 seconds. Therefore, when we attempted to synthesize longer utterances (specifically the full paragraphs from NPR stories used for testing our Merlin models), DCTTS outputted very poor quality audio. In the future, we would like to re-train DCTTS using a training set with longer utterances. While there are relatively few TTS corpora segmented at the paragraph level, it may be possible to re-segment LJSpeech into longer utterances, as the data in LJSpeech originally comes from audiobooks. Therefore, it may be possible to re-train DCTTS on this re-segmented data with few other changes to our model. Other audiobook corpora may also be useful for these purposes. In particular, LibriTTS [67] is a very large corpus of audiobook data. Like LJSpeech and most other TTS corpora, it has been segmented at the sentence level, but it contains more information about the original audiobooks, possibly making the resegmentation process easier.

Finally, it has yet to be seen if any of these models will generalize to conversational speech. So far, we have only trained our models on read speech, which makes it hard to replicate the prosody of conversational speech (which, as our poor cross-genre results for prosody prediction showed, is very different from read speech). However, in the future, I plan to train a model directly on the Switchboard Corpus. Since the entire Switchboard Corpus has over 200 hours of speech, it should be possible to remove noisy utterances and short utterances and still have enough data to train an end-to-end system. Unfortunately, until recently, there was no good way to accurately segment the corpus, as the utterance-level alignments provided with previous versions of the corpus contained a large number of errors. Currently, there is ongoing work to fix a large number of these alignments (specifically the alignments for conversations within the Switchboard Dialog Act Corpus [57]), which will hopefully make training on Switchboard feasible. Additionally, since a large amount of Switchboard contains gold standard dialog act labels, training on Switchboard would allow us to determine if information about dialog acts could be used to improve the prosody of synthesized speech. While we found that including dialog act features did not improve the prosody prediction models trained on Switchboard, this is likely due to the fact that dialog acts have little effect on phrase breaks and pitch accent but have a much larger effect on sentence-level prosody. Therefore, incorporating dialog act labels directly into an end-to-end framework to condition sentence-level prosody has potential as a future research direction.

5.8 Conclusion

In this chapter, I have shown that we can improve the output of DCTTS, a convolutional end-to-end TTS system, by including the locations of phrase boundaries in the input text. We trained this model on LJSpeech, a commonly used TTS corpus of read audiobook speech. In order for our model to learn the acoustic properties of phrase boundaries, we used AuToBI, a system that automatically predicts ToBI labels from text, to find the locations of phrase boundaries in LJSpeech. We then used our text-based model to predict the locations of phrase boundaries in sentences from NPR broadcasts and synthesized these sentences using our trained DCTTS model. We also synthesized a set of sentences taken from LJSpeech, using AuToBI as an "oracle" system to place phrase boundaries..

We compared the sentences from both test sets synthesized using phrase boundaries to the same sentences synthesized using a baseline version of DCTTS trained on LJSpeech. In both cases, we calculated the MCD between our synthesized sentences and the audio from the original LJSpeech Corpus or NPR broadcasts. In both cases, we found that it was lower than the MCD between the baseline synthesized sentences and the original audio. However, when we ran a listening test using the synthesized test utterances, we found that listener preference for the modified voice was not statistically significant, and preferences varied depending on the utterance. These results are therefore inconclusive, though they indicate that incorporating prosodic events into an end-to-end TTS pipeline may be able improve the synthesized speech.

These preliminary results are encouraging and show that it is worth pursuing future work on incorporating prosodic events into end-to-end synthesis. In particular, adding pitch accents, as we did in our Merlin pipeline, could further improve the prosody. It also remains to be seen whether this approach can be used to synthesize longer utterances and, beyond that, whether it is possible to use a similar approach with Switchboard data in order to synthesize conversational speech with appropriate prosody.

Conclusion

In this thesis, I have presented work focused on predicting and producing prosodic events in order to improve speech synthesis. In chapters 2 and 3, I presented new models for predicting prosodic events, specifically phrase boundaries and pitch accents, from text using a large variety of linguistic features. These chapters included models trained on radio news data and conversational data, including both task-oriented speech and open-ended conversational speech. In chapter 4, I showed that it is possible to incorporate these prosodic events into a a TTS pipeline using the Merlin toolkit to produce natural speech that replicates the prosody of radio news. Finally, in chapter 5, I presented an adaptation of this approach for speech synthesis using the end-to-end DCTTS model. This model was trained using a corpus of read speech, with phrase boundaries predicted from audio features, and was able to synthesize sentences from radio news broadcasts, using phrase boundaries predicted from text features.

The work presented in chapters 2 and 3 represents a significant contribution to the area of prosody prediction from text. First, we achieve much higher accuracy on specifically the phrase boundary detection task than previous models from the literature trained on the same corpora. Second, we present models that use a broader variety of linguistic features than all previous work in this area, including some features that have never been used before for these tasks. For example, we found that the level of specificity in a sentence was a helpful feature in our models for all corpora on both tasks. Finally, by using interpretable models trained on a large variety of linguistic features, we are able to better understand which linguistic features have the strongest effects on pitch accent and phrasing. For example, we found that word embeddings were extremely helpful for our models, particularly on conversational corpora and particularly for the pitch accent detection task. However, we found that on pitch accent models for BURNC, gender-neutral embeddings trained on Wikipedia were more useful than embeddings trained on Google News (for BURNC). These results indicate that some semantic information is useful for predicting prosody, but that fine-grained semantic information (such as information about gender) is not. Additionally, we found that co-reference features were helpful for predicting pitch accent on corpora, but we also found that the only minorly improved performance. This result is particularly notable on the Games Corpus, which contains task-oriented speech specifically intended to capture the difference in accent between given and new information. These results may indicate that the effect of information status on pitch accent, while existent, is not particularly pronounced.

In chapter 4, I demonstrated that phrase boundaries and pitch accents can be directly included as features to text-to-speech system to improve the prosody of synthesized speech. This improvement was particularly striking when synthesizing longer utterances, as we found that 80% of listeners preferred paragraphs synthesized with a model using a prosodically enhanced pipeline as compared to the default Merlin pipeline. In chapter 5, I showed that that we can get similar results using an end-to-end TTS pipeline and including phrase boundaries into the input. We found that by training DCTTS on data that included phrase boundaries we were able to better replicate radio news speech than we could using the default DCTTS pipeline. The results in these two chapters indicate that directly incorporating prosodic events is a viable approach for improving the prosody of synthesized speech. This approach stands in contrast to most current work in this area, which instead largely focuses on controlling prosody at the utterance level or on incorporating linguistic features that indirectly affect prosody (such as syntactic features).

While the work presented in this thesis substantially contributes to its field, there is still much work to do, particularly in further exploring how prosodic events can improve end-to-end TTS. Future work will focus on incorporating pitch accents into an end-to-end pipeline and on synthesizing longer utterances with that pipeline. Additionally, there is still a large amount of work on synthesizing speech that replicates to prosody of conversational speech. While I was able to build accurate models for predicting prosodic events in conversational speech, I have not yet been able to synthesize conversational speech. In the future, I plan to train TTS systems on the Switchboard Corpus while incorporating prosodic features. Additionally, these systems will allow me to further explore the use of linguistic features specific to conversational speech, such as dialog acts. Overall, I have presented a promising approach to prosody modeling, but there is still work to be done, especially in the conversational domain.

References

[1] R. Sloan, S. S. Akhtar, B. Li, R. Shrivastava, A. Gravano, and J. Hirschberg, "Prosody prediction from syntactic, lexical, and word embedding features," in *10th ISCA Speech Synthesis Workshop*, 2019.

[2] R. Sloan, A. Adigwe, S. Mohandoss, and J. Hirschberg, "Incorporating prosodic events in text-to-speech synthesis," *Proc. Speech Prosody 2022*, pp. 287–291, 2022.

[3] M. E. Beckman and J. Hirschberg, "The ToBI annotation conventions," *Ohio State University*, 1994.

[4] A. Rosenberg, "AuToBI – A tool for automatic ToBI annotation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[5] J. Hirschberg and P. Prieto, "Training intonational phrasing rules automatically for English and Spanish text-to-speech," *Speech Communication*, vol. 18, no. 3, pp. 281–290, 1996.

[6] K. Ross and M. Ostendorf, "Prediction of abstract prosodic labels for speech synthesis," *Computer Speech & Language*, vol. 10, no. 3, pp. 155–185, 1996.

[7] J. Hirschberg, "Pitch accent in context predicting intonational prominence from text," *Artificial Intelligence*, vol. 63, no. 1-2, pp. 305–340, 1993.

[8] J. Hirschberg and O. Rambow, "Learning prosodic features using a tree representation," in *Seventh European Conference on Speech Communication and Technology*, 2001.

[9] P. Koehn, S. Abney, J. Hirschberg, and M. Collins, "Improving intonational phrasing with syntactic information," in *2000 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, vol. 3, 2000, pp. 1289–1290.

[10] T. Ingulfsen, "Influence of syntax on prosodic boundary prediction," University of Cambridge, Computer Laboratory, Tech. Rep., 2004.

[11] K. Chen, M. Hasegawa-Johnson, and A. Cohen, "An automatic prosody labeling system using ann-based syntactic-prosodic model and gmm-based acoustic-prosodic model," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, IEEE, vol. 1, 2004, pp. I–509.

[12] J. Tepperman and E. Nava, "Where should pitch accents and phrase breaks go? A syntax tree transducer solution," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[13] N. Obin and P. Lanchantin, "Symbolic modeling of prosody: From linguistics to statistics," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 588–599, 2015.

[14] T. Mishra, Y.-j. Kim, and S. Bangalore, "Intonational phrase break prediction for text-to-speech synthesis using dependency relations," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 4919–4923.

[15] J. Yuan, J. M. Brenier, and D. Jurafsky, "Pitch accent prediction: Effects of genre and speaker," in *Ninth European Conference on Speech Communication and Technology*, 2005.

[16] A Nenkova, J Brenier, A Kothari, S Calhoun, L Whitton, D Beaver, and D Jurafsky, "To memorize or to predict: Prominence labeling in conversational speech," in *Proceedings of NAACL HLT*, 2007, pp. 9–16.

[17] J. M. Brenier, A. Nenkova, A. Kothari, L. Whitton, D. Beaver, and D. Jurafsky, "The (non) utility of linguistic features for predicting prominence in spontaneous speech," in *2006 IEEE Spoken Language Technology Workshop*, IEEE, 2006, pp. 54–57.

[18] A. Rendel, R. Fernandez, R. Hoory, and B. Ramabhadran, "Using continuous lexical embeddings to improve symbolic-prosody prediction in a text-to-speech front-end," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 5655–5659.

[19] A. Vadapalli and S. V. Gangashetty, "An investigation of recurrent neural network architectures using word embeddings for phrase break prediction.," in *Proceedings of Interspeech 2016*, 2016, pp. 2308–2312.

[20] V. Klimkov, A. Nadolski, A. Moinet, B. Putrycz, R. Barra-Chicote, T. Merritt, and T. Drugman, "Phrase break prediction for long-form reading tts: Exploiting text structure information," in *Proceedings of Interspeech 2017*, 2017, pp. 1064–1068.

[21] Y. Zheng, J. Tao, Z. Wen, and Y. Li, "Blstm-crf based end-to-end prosodic boundary prediction with context sensitive embeddings in a text-to-speech front-end," in *Proceedings of Interspeech 2018*, 2018, pp. 47–51.

[22] A. Rendel, R. Fernandez, Z. Kons, A. Rosenberg, R. Hoory, and B. Ramabhadran, "Weakly-supervised phrase assignment from text in a speech-synthesis system using noisy labels," in *Proceedings of Interspeech 2017*, 2017, pp. 759–763.

[23] V. Soto, E. Cooper, A. Rosenberg, and J. Hirschberg, "Cross-language phrase boundary detection," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2013, pp. 8460–8464.

[24] A. Rosenberg, R. Fernandez, and B. Ramabhadran, "Phrase boundary assignment from text in multiple domains," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[25] T. Weijters and J. Thole, "Speech synthesis with artificial neural networks," in *IEEE International Conference on Neural Networks*, IEEE, 1993, pp. 1764–1769.

[26] C. Tuerk and T. Robinson, "Speech synthesis using artificial neural networks trained on cepstral coefficients," in *Third European Conference on Speech Communication and Technology*, 1993.

[27] H. Ze, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *2013 ieee international conference on acoustics, speech and signal processing*, IEEE, 2013, pp. 7962–7966.

[28] H. Lu, S. King, and O. Watts, "Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis," in *Eighth ISCA workshop on speech synthesis*, 2013.

[29] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, "On the training aspects of deep neural network (dnn) for parametric tts synthesis," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 3829–3833.

[30] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 4470–4474.

[31] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks," in *Fifteenth annual conference of the international speech communication association*, 2014.

[32] Z. Wu, O. Watts, and S. King, "Merlin: An open source neural network speech synthesis system.," in *SSW*, 2016, pp. 202–207.

[33] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, *et al.*, "Tacotron: Towards end-to-end speech synthesis," *Proc. Interspeech 2017*, pp. 4006–4010, 2017.

[34] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4779–4783.

[35] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, pp. 125–125.

[36] Z. Malisz, H. Berthelsen, J. Beskow, and J. Gustafson, "Controlling prominence realisation in parametric dnn-based speech synthesis," in *Proceedings of Interspeech 2017*, 2017, pp. 1079–1083.

[37] T. Fujimoto, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Impacts of input linguistic feature representation on japanese end-to-end speech synthesis," in *10th ISCA Speech Synthesis Workshop. ISCA, Vienna, Austria*, 2019.

[38] H. Guo, F. K. Soong, L. He, and L. Xie, "Exploiting syntactic features in a parsed tree to improve end-to-end tts," *arXiv preprint arXiv:1904.04764*, 2019.

[39] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "Word embedding for recurrent neural network based tts synthesis," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 4879–4883.

[40] Y. Xiao, L. He, H. Ming, and F. K. Soong, "Improving prosody with linguistic and bert derived features in multi-speaker based mandarin chinese neural tts," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 6704–6708.

[41] P. O. Gallegos, J. O'Mahony, and S. King, "Comparing acoustic and textual representations of previous linguistic context for improving text-to-speech," in *The 11th ISCA Speech Synthesis Workshop (SSW11)*, ISCA, 2021, pp. 205–210.

[42] Y. Wang, D. Stanton, Y. Zhang, R.-S. Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 5180–5189.

[43] S. Tyagi, M. Nicolis, J. Rohnke, T. Drugman, and J. Lorenzo-Trueba, "Dynamic Prosody Generation for Speech Synthesis Using Linguistics-Driven Acoustic Embedding Selection," in *Proc. Interspeech 2020*, 2020, pp. 4407–4411.

[44] M. Ostendorf, P. J. Price, and S. Shattuck-Hufnagel, "The Boston University radio news corpus," *Linguistic Data Consortium*, pp. 1–19, 1995.

[45] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60.

[46] J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn, "The development and psychometric properties of LIWC2015," The University of Texas at Austin, Tech. Rep., 2015.

[47] A. K. Joshi and B. Srinivas, "Disambiguation of super parts of speech (or supertags): Almost parsing," in *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, ser. COLING '94, Kyoto, Japan: Association for Computational Linguistics, 1994, pp. 154–160.

[48] J. Kasai, B. Frank, T. McCoy, O. Rambow, and A. Nasr, "Tag parsing with neural networks and vector representations of supertags," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1712–1722.

[49] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[50] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[51] H. Gonen and Y. Goldberg, "Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them," *arXiv preprint arXiv:1903.03862*, 2019.

[52] J. J. Li and A. Nenkova, "Fast and accurate prediction of sentence specificity.," in *AAAI*, 2015, pp. 2281–2287.

[53] A. Gravano and J. Hirschberg, "Turn-taking cues in task-oriented dialogue," *Computer Speech & Language*, vol. 25, no. 3, pp. 601–634, 2011.

[54] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, IEEE Computer Society, vol. 1, 1992, pp. 517–520.

[55] S. Calhoun, J. Carletta, J. M. Brenier, N. Mayo, D. Jurafsky, M. Steedman, and D. Beaver, "The nxt-format switchboard corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue," *Language resources and evaluation*, vol. 44, no. 4, pp. 387–419, 2010.

[56] S. S. Akhtar, A. Gupta, A. Vajpayee, A. Srivastava, M. G. Jhawar, and M. Shrivastava, "An unsupervised approach for mapping between vector spaces," *arXiv preprint arXiv:1711.05680*, 2017.

[57] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer, "Dialogue act modeling for automatic tagging and

recognition of conversational speech," *Computational linguistics*, vol. 26, no. 3, pp. 339–373, 2000.

[58] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, "The hmm-based speech synthesis system (hts) version 2.0.," in *SSW*, Citeseer, 2007, pp. 294–299.

[59] A. Black, P. Taylor, R. Caley, and R. Clark, *The festival speech synthesis system*, 1998.

[60] O. Watts, G. E. Henter, J. Fong, and C. Valentini-Botinhao, "Where do the improvements come from in sequence-to-sequence neural tts?" In *2019 ISCA Speech Synthesis Workshop (SSW)*, vol. 10, 2019, pp. 217–222.

[61] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[62] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4784–4788.

[63] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," *Advances in neural information processing systems*, vol. 32, 2019.

[64] tugstugi, *Text to speech with pytorch (english and mongolian)*, 2018.

[65] K. Ito, *The lj speech dataset*, 2017.

[66] A. Pettarin, *Aeneas*, 2017.

[67] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "Libritts: A corpus derived from librispeech for text-to-speech," *Proc. Interspeech 2019*, pp. 1526–1530, 2019.

Appendix A: List of Features

A full list of features used for training the models is presented below. During training, similar features were grouped together (e.g. all syntactic features formed a group). This grouping method allowed us to iterate over possible combinations of groups of features to find the optimal model. Therefore, in the following lists, features have been listed according to their groups. Additionally, please note that some features that take on categorical features (such as POS tag) were automatically converted into binary features during the training process, and each binary feature had its own feature weight after training. Here, I have only listed the original features before they were converted into binary features. I have also included features that were never included in any of the optimal models but were explored during the training process.

A.1 BURNC Models

A.1.1 Syntactic and Part of Speech

The following features relate to syntax (taken either from the constituency or dependency parse) or part of speech. Unless otherwise noted, any references to tree refer to the constituency parse of the current sentence, any references to constituent refer to the smallest nontrivial constituent containing the word, and any references to the spanning tree refer to the minimal spanning tree between the current word and the next word.

Features: syntactic function, next word's syntactic function, word depth in tree, tree depth, tree width, width of constituent, label of constituent, forward position in constituent, backward position in constituent, depth of spanning tree, width of spanning tree, spanning tree label, part of speech tag, next word's part of speech tag.

A.1.2 Supertag

In order to test if supertags were useful, the supertag feature was in a group by itself.

A.1.3 Punctuation

The punctuation feature group contained only one feature: punctuation after the current word.

A.1.4 Position

The positional features group includes three features: length of sentence, word's forward position in sentence, and word's backward position in sentence.

A.1.5 Syllables

The number of syllables in the current word was in a group by itself.

A.1.6 Named Entity Recognition

The NER group contained two features: current words's named entity recognition label and the next word's named entity recongition label.

A.1.7 Mentions

The mentions group contained a number of features related to co-reference and previous mentions. The features are listed below.

Features: number of previous mentions, number of previous explicit mentions, number of previous implicit mentions, most recent mention's POS, most recent explicit mention's POS, most recent implicit mention's POS, most recent mention's syntactic function, most recent explicit mention's syntactic function, most recent implicit mention's syntactic function, distance to most recent mention

A.1.8 LIWC

Each LIWC feature was a binary feature corresponding to a LIWC category. A full list of LIWC categories used is listed below.

Features: religion, body, comparison, money, pronoun, feeling, certainty, sexual, insight, assent, number, sad, time, anger, cognitive processes, female, article, negation, home, conjunction, anxiety, negative emotion, personal pronoun, difference, death, family, adverb, space, informal, first person pronoun, present tense, netspeak, perception, quantity, discrepancy, relativity, health, second person pronoun, adjective, preposition, friendship, function word, biology, we pronoun, risk, power, interrogation, social, drives, leisure, verb, hearing, past tense, they pronoun, affect, nonfluency, tentative, reward, affiliation, I pronoun, cause, work, seeing, ingestion, motion, filler word, swear, achievement, future tense, positive emotion, auxiliary verb, male, she or he pronoun

A.1.9 Embeddings

Each of the types of embeddings used for BURNC had its own group. This led to five groups: embeddings trained on BURNC, embeddings domain-adapted to BURNC, pre-trained Google news embeddings, pre-trained dependency embeddings, and pre-trained gender-neutral embeddings. Each of these groups has 14 features. In all cases, all features were cluster IDs for the current word, the surrounding words, the current sentence, and the surrounding sentences

Features: current word embedding, previous word embedding, second previous word embedding , third previous word embedding , next word embedding, second next word embedding, third next word embedding, current sentence embedding, previous sentence embedding, second previous sentence embedding, third previous sentence embedding, next sentence embedding, second next sentence embedding, third next sentence embedding

A.1.10 Speciteller

The Speciteller category included only one feature, which corresponded to the current sentence's Speciteller score.

A.2 Games Models

A.2.1 Syntactic and Part of Speech

The syntax and part of speech group for Games contained the same set of features used in BURNC.

A.2.2 Supertag

Again, the supertag feature was in a group by itself.

A.2.3 Position

For Games, we used a longer set of positional features, accounting for position in turn and task, as well as sentence.

Features: length of sentence, word's forward position in sentence, word's backward position in sentence, length of turn, word's forward position in turn, word's backward position in turn, length of task, word's forward position in task, word's backward position in task

A.2.4 Syllables

Again, the number of syllables in the current word was in a group by itself.

A.2.5 Mentions

The set of co-reference features used in Games was the same as the set used for BURNC.

A.2.6 LIWC

The same set of LIWC categories were used for all corpora.

A.2.7 Embeddings

We looked at three different types of embeddings for Games: pre-trained embeddings based on Twitter, domain-adapted embeddings, and gender-neutral embeddings. The domain-adapted embeddings and gender-neutral embeddings each had their own groups, each of which contained the same 14 features that they did in the BURNC models.

For the pre-trained Twitter embeddings, we had two groups: one consisting of only embeddings of the current word, the other consisting of the embeddings for all surrounding words, as well as all sentence embeddings. This group, named the neighboring embeddings group, therefore contained the same set of features as all of the BURNC groups. We make this split for two reasons. First, as embeddings were highly useful when training games, this allowed us to see the impact that incorporating the neighboring words' embeddings had on model performance. Second, there were several different GLoVE models pre-trained on Twitter, each of whi had a different dimension. For the neighboring embeddings, we used only the highest dimensional embeddings (200-dimensional embeddings), but for the current word's embedding we also included cluster IDs for several smaller embeddings. These include 25-dimensional embeddings, 50-dimensional embeddings, 100-dimensional embeddings, and 200-dimensional embeddings.

A.2.8 Speciteller

Again, the Speciteller category included only one feature, the current sentence's Speciteller score.

A.2.9 Durational

The durational features category was used only for a small set of experiments. It included five features: number of words in IPU, word's forward position in IPU, word's backward position in IPU, length of silence before word, and length of sentence after word.

A.3 Switchboard

The features used when training Switchboard are largely the same as those used to train Games. Therefore, instead of specifying every feature group used during training, I will discuss broader feature categories and mention where the Switchboard features vary from Games.

A.3.1 Syntax

The Switchboard model uses all features from the syntax/part of speech group from Games, as well as supertags. Additionally, Switchboard's model uses a number of additional syntactic features that were not used with either other corpus. These features are listed below.

New syntax features: highest-level phrase ending with the preceding word, highest-level phrase beginning with the current word lowest-level node, distance in the tree between current and next word, whether current word is at the right boundary of a major constituent, whether current word is at the right boundary of a coordinating constiutent, size of subtree headed by current word, number of siblings of current word, POS tag for previous word, POS tag for second previous word, POS tag for second next word

A.3.2 Positional

The Switchboard model included the positional features in Games, excluding those related to the current task (as Switchboard is not task-oriented). Additionally, we added a relative position in sentence features.

A.3.3 Embeddings

For Switchboard, we used only the GLoVE embeddings pre-trained on Twitter. Therefore, we used both the Twitter embeddings and neighboring embeddings groups from Games (but no other embedding features).

A.3.4 Other

The co-reference, LIWC, Speciteller and syllables groups from the Games models were used unchanged. Additionally, this model included the two NER features from BURNC (due to the fact that Switchboard contains named entities but Games largely does not).

Appendix B: Full BURNC and Games Results

This appendix provides a fuller set of results for the prosody prediction models trained on BURNC and on Games, discussed in chapters 2 and 3. In particular, this appendix includes tables demonstrating the impact of removing each feature group from the best models on each task. Additionally, I have provided a list of the 20 most heavily weighted features (according to the random forest model) on each task.

B.1 BURNC

B.1.1 Impact of Feature Groups

Feature Set	**Accuracy**	**F1**	**Precision**	**Recall**
Best Model	93.4%	0.810	0.810	0.810
Best Model without NER	93.2%	0.798	0.812	0.785
Best Model without Embeddings	92.9%	0.792	0.798	0.785
Best Model without Speciteller	92.5%	0.782	0.779	0.785
Best Model without Punctuation	92.5%	0.774	0.798	0.752
Best Model without Supertag	92.0%	0.763	0.783	0.744
Best Model without Syllables	92.0%	0.770	0.764	0.777
Best Model without Syntax	85.5%	0.575	0.580	0.570
Punctuation Only Baseline	91.2%	0.690	0.873	0.570
Majority Class Baseline	82.8%			

Table B.1: Performance for **phrase boundary** detection on BURNC, displaying the impact of each feature group

Tables B.1 and B.2 display the effects of removing each feature group from the best models for phrase boundary and pitch accent detection on BURNC, respectively. The best model for phrase boundaries has 7 feature groups (as discussed in the previous appendix): syntax, punctuation, syllables in word, named entity recognition, supertags, word embeddings (trained directly on BURNC),

Feature Set	Accuracy	F1	Precision	Recall
Best Model	81.9%	0.844	0.776	0.925
Best Model Without Supertag	81.6%	0.841	0.776	0.916
Best Model Without Mentions	81.5%	0.840	0.774	0.916
Best Model without Speciteller	81.3%	0.838	0.773	0.916
Best Model without Punctuation	81.2%	0.837	0.771	0.916
Best Model without Syllables	81.1%	0.836	0.770	0.914
Best Model without LIWC	80.9%	0.832	0.778	0.895
Best Model without NER	80.8%	0.834	0.768	0.911
Best Model without Embeddings	80.2%	0.826	0.771	0.889
Best Model without Syntax	79.1%	0.817	0.758	0.887
Content Words Baseline	79.9%	0.823	0.771	0.881
Majority Class Baseline	52.8%			

Table B.2: Performance for **pitch accent** detection on BURNC, displaying the impact of each feature group

and Speciteller. The best model for pitch accents has 9 feature groups: syntax, punctuation, syllables in word, named entity recognition, supertag, mentions, LIWC, gender-neutral embeddings, and Speciteller.

All models are listed from highest accuracy to lowest accuracy, meaning that the most impactful feature groups are listed near the bottom of each table. (As all values have been rounded to three significant digits, in some cases, it may appear that two features had exactly the same effect on accuracy, when there was in fact a very small but existent difference. As the accuracies were originally calculated to more significant digits, the tables are in fact strictly ordered from highest to lowest accuracy, excluding the baselines, which are always listed at the bottom.) Additionally, note that in some cases, models with higher accuracy have lower F1 scores. The performance of the best model, as well as two baselines for each task, are shown for reference.

B.1.2 Feature Importances

Here, I list the top 20 most important features for each of the best models on BURNC, as determined by the random forest training algorithm. For both lists, features are ranked from most important to least important.

The random forest algorithm does not directly compute feature weights as part of the training process. However, it is possible to calculate the importance of feature by averaging the impurity decrease across all nodes where a feature is used across all trees in the final model. There are some drawbacks to this approach. (For example, it often prioritizes continuous features over discrete features.) However, in general, this method gives a strong sense of which individual features had the greatest predictive power. Additionally, for features that take on categorical values, such as POS tag, this approach indicates which particular tags were particularly helpful.

Top 20 Features for Phrase Boundary Detection

1. Width of minimal spanning tree with next word
2. Speciteller score
3. Width of constituency parse
4. Word depth in constituency parse
5. Backward position in smallest constituent
6. Depth of minimal spanning tree with next word
7. Depth of constituency parse
8. Supertag t3
9. Number of syllables in word
10. Followed by period
11. Forward position in smallest constituent
12. Width of smallest constituent
13. Followed by comma

14. Minimal spanning tree with next word has label S

15. POS tag NN

16. POS tag NNS

17. Syntactic function nsubj

18. Smallest constituent has label NP

19. Syntactic function dobj

20. Minimal spanning tree with next word has label NP

Top 20 Features for Pitch Accent Detection

1. LIWC function word category

2. Number of syllables in word

3. Speciteller score

4. Width of constituency parse

5. Next sentence gender-neutral embedding

6. Depth of constituency parse

7. Second next sentence gender-neutral embedding

8. Previous sentence gender-neutral embedding

9. Second previous sentence gender-neutral embedding

10. Current sentence gender-neutral embedding

11. Third next gender-neutral embedding

12. Word depth in constituency parse
13. Third previous sentence gender-neutral embedding
14. Width of minimal spanning tree with next word
15. Width of smallest constituent
16. Backward position in smallest constituent
17. Depth of minimal spanning tree with next word
18. Forward position in smallest constituent
19. LIWC article category
20. Smallest constituent has label PP

B.2 Games

B.2.1 Impact of Feature Groups

Feature Set	**Accuracy**	**F1**	**Precision**	**Recall**
Best Model	90.5%	0.774	0.890	0.687
Best Model Without Speciteller	90.5%	0.773	0.888	0.687
Best Model Without Mentions	90.5%	0.773	0.890	0.686
Best Model Without Gender-Neutral Embeddings	90.5%	0.773	0.888	0.686
Best Model Without LIWC	90.5%	0.772	0.892	0.681
Best Model Without Syllables	90.4%	0.772	0.887	0.685
Best Model Without Domain-Adapted Embeddings	90.4%	0.771	0.887	0.684
Best Model Without Neighboring Embeddings	90.4%	0.771	0.883	0.686
Best Model Without Embeddings	90.4%	0.769	0.894	0.676
Best Model Without Syntax	90.1%	0.761	0.884	0.670
Best Model Without Positional	90.0%	0.764	0.867	0.686
Positional Features Only	88.3%	0.711	0.867	0.603
Majority Class Baseline	76.3%			

Table B.3: Performance scores for **phrase boundary** detection on Games, displaying the effects of all feature groups

Feature Set	**Accuracy**	**F1**	**Precision**	**Recall**
Best Model	82.1%	0.818	0.814	0.827
Best Model Without Speciteller	82.1%	0.818	0.814	0.827
Best Model Without Supertag	82.1%	0.818	0.814	0.826
Best Model Without Mentions	82.1%	0.818	0.814	0.825
Best Model Without Positional	82.0%	0.818	0.813	0.826
Best Model Without Gender-Neutral Embeddings	82.0%	0.818	0.813	0.827
Best Model Without Syllables	82.0%	0.817	0.813	0.826
Best Model Without LIWC	82.0%	0.817	0.812	0.825
Best Model Without Neighboring Embeddings	81.8%	0.816	0.811	0.825
Best Model Without Embeddings	81.7%	0.815	0.806	0.829
Best Model Without Syntax	81.7%	0.813	0.816	0.814
Content Words Baseline	75.0%	0.746	0.742	0.750
Majority Class Baseline	51.1%			

Table B.4: Performance scores for **pitch accent** detection on Games, displaying the effects of all feature groups

Tables B.3 and B.4 display the effects of removing each feature groups from the best models for phrase boundary prediction and pitch accent detection respectively on the Games Corpus. Due to the cross-validation scheme for training and testing on Games, discussed in chapter 3, all metrics were calculated by aggregating over all 12 test folds. (Since each test speaker spoke a different number of words, this was done by taking a weighted average across all folds.) Each model used 10 feature groups. Both models used syntax, positional features, mentions, syllables, GLoVe embeddings, neighboring embeddings, gender-neutral embeddings, LIWC, and Speciteller. Additionally, the phrase model used domain-adapted embeddings, and the accent model used supertags.

For both tables, models have been sorted from highest accuracy to lowest accuracy, Again, there are cases where two models appear to have equal accuracy when rounded to three significant digits. For example, removing the mentions feature group does decrease both accuracy and F1 on both models, but the effects are very slight (and not clearly visible in this chart, due to rounding effects). Similarly, some models may appear to have the same precision and recall but different F1 scores due to rounding effects. The order of the tables means that models lower in the table are the models where more impactful features were removed (with the caveat that higher accuracy does not always correspond to higher F1 score). Again, the best model, along with two baselines for each task, has been provided for reference.

B.2.2 Feature Importances

Here are the top 20 features for the phrase boundary detection and pitch accent models on the Games Corpus, as determined by the random forest model. In this case, due to the cross-validation process used to test Games, we have had to determine the most important features aggregating across all 12 folds. This was done by averaging each feature's importance across all 12 models, then reranking using the averaged importances. Feature importance rankings for each individual fold are not displayed here, and while the feature rankings were not identical for each fold, they were fairly similar. (For example, words until end of sentence was either the first or second most important feature for the phrasing model on all folds.) The features are listed, beginning with the most important features.

Top 20 Features for Pitch Accent Detection

1. Backward position in sentence
2. Minimal spanning tree with next word has label ROOT
3. Backward position in turn
4. Backward position in smallest constituent
5. Backward position in task
6. Next word's GLoVe embedding
7. Total number of words in sentence
8. Width of constituency parse
9. Width of minimal spanning tree with next word
10. Total number of words in turn

11. Depth of constituency parse
12. Depth of minimal spanning tree with next word
13. Speciteller score
14. Total number of words in sentence
15. Forward position in task
16. Forward position in turn
17. Width of smallest constituent
18. Word depth in constituency parse
19. Forward position in sentence
20. POS tag NN

Top 20 Features for Pitch Accent Detection

1. LIWC function word category
2. GLoVe Twitter 25-dimensional embedding
3. GLoVe Twitter 200-dimensional embedding
4. Number of syllables in word
5. GLoVe Twitter 50-dimensional embedding
6. GLoVe 200-dimensional embedding, trained on 6 billion words
7. Total number of words in task
8. Width of constituency parse

9. Forward position in task
10. Backward position in task
11. Speciteller score
12. Total number of words in term
13. Backward position in turn
14. Forward position in task
15. GLoVe Twitter 100-dimensional embedding
16. Depth of constituency parse
17. Word depth in constituency parse
18. Width of minimal spanning tree with next word
19. Backward position in smallest constituent
20. Second next sentence embedding